AFRL-RI-RS-TR-2012-115

DISTRIBUTED EPISODIC EXPLORATORY PLANNING (DEEP)

APRIL 2012

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2012-115 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
ALEX SISTI
Chief, Advanced Planning &
Autonomous C2 Systems Branch

/s/
JULIE BRICHACEK, Chief
Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
APR 2012	Final Technical Report	JUL 2006 – SEP 2011

4. TITLE AND SUBTITLE

DISTRIBUTED EPISODIC EXPLORATORY PLANNING (DEEP)

5a. CONTRACT NUMBER
In House

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Dale Richards, Kurt Lachevet, Gennady Staskevich,
Anthony Ford, Chad DeStefano

5d. PROJECT NUMBER
558S

5e. TASK NUMBER
IH

5f. WORK UNIT NUMBER
DP

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/RISC
525 Brooks Road
Rome, NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site /RISC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2012-115

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. **PA# 88ABW-2012-1887**
Date Cleared: **30 MAR 12**

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report presents an overview of the work performed on the Distributed Episodic Exploratory Planning (DEEP) project. DEEP is a mixed-initiative decision support system that utilizes past experiences to suggest courses of action for new situations. It has been designed as a distributed multi-agent system, using agents to maintain and exploit the experiences of individual commanders, as well as to transform suggested past plans into potential solutions for a new situation or world state. The system is mixed-initiative in the sense that a commander, through his or her agent, can view and modify the contents of the shared repository as needed. The agents interact through a common knowledge repository and managed by a blackboard architecture. The blackboard architecture is well suited for dealing with ill-described, complex situations that are part of modern warfare. Additional aspects of the work include development of a common plan representation and design of the case repository feature space. Much of the work has been documented in numerous other papers, reports and presentations, as referenced in the bibliography, and is only summarized in this report.

15. SUBJECT TERMS

Planning, Case Based Reasoning, Mixed-Initiative, Blackboard Systems, Agent Based Systems, Robust Coherence, Course of Action, COA

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dale W. Richards
U	U	U	UU	60	19b. TELEPHONE NUMBER (*Include area code*) N/A

Table of Contents

List of Figures

1. Executive Summary

This report describes research done under the Distributed Episodic Exploratory Planning (DEEP) project. DEEP is a mixed-initiative decision support system that utilizes past experiences to suggest courses of action for new situations. It has been designed as a distributed multi-agent system, using agents to maintain and exploit the experiences of individual commanders as well as to transform suggested past plans into potential solutions for new problems. The system is mixed-initiative in the sense that a commander, through his or her agent, can view and modify the contents of the shared repository as needed. The software agents interact through a common knowledge repository, managed by via a blackboard-based architecture. The blackboard architecture is well suited for dealing with uncertain, complex situations that are part of modern warfare. Additional aspects of the work include development of a common plan representation and design of the case repository feature space. Much of the work is documented in detail in other papers, reports and presentations, as referenced in the bibliography, and is only summarized in this report.

DEEP was initiated in 2006 in response to command and control (C2) deficiencies with regards to development of a course of action (COA) satisfying the stated and implied conditions embodied in a given commander's intent. A means for providing mixed-initiative support to a planning staff -- potentially distributed in time, space and network presence – was needed as part of the push toward network centric operations (NCO). This project sought to develop, in-house, a research platform for exploring distributed, mixed-initiative planning using analogical reasoning over an experience base of past actions.

The resulting initial research platform is comprised of the following components:

- *Distributed Blackboard* to support multi-agent, non-deterministic, opportunistic reasoning
- *Case-Based Reasoning* to capture experiences (successes and/or failures)
- *ARPI Core Plan Representation* (CPR) for human-to-machine common dialog
- *Multi-Agent System* for mixed initiative planning

Additional components associated with the comprehensive vision were planned, but not implemented. These include:

- *Episodic Memory* for powerful analogical reasoning
- *Constructive Simulation* for exploration of possible future states

Research areas subsequently investigated using the DEEP platform include:

- *Distributed Episodic Analogical Reasoning* to increase system robustness (DEAR) (Mulvehill, et al., 2007)

- *Semantic Extensions to the CPR* to provide a richer plan representation
- *Robust Coherence* to mediate and integrate output from multiple agents
- *Constraint Satisfaction* to align past plans with current needs
- *Model Adaptation* to autonomously maintain model relevance
- *Mixed-Initiative Critic Agents* to critique and adapt retrieved plans
- *Real-time Strategy Game Replays* as extensive case bases

The research initiated under this program is being utilized as a foundation for new research tasks scheduled to begin in 2012. The primary emphasis of this work will focus on dynamically managing the case retrieval process by selecting the optimal set of case features on which to base retrieval of plans and plan fragments from the case base, as well as an assessment of the degree to which execution of a plan has strayed from an earlier, anticipated path to determine when a plan should be monitored, adapted or re-planned. The DEEP platform will continue to be used to support exploration of case-based reasoning for plan retrieval and course-of-action development.

2. Introduction

DEEP is a mixed-initiative decision support system that utilizes past experiences to suggest courses of action for new situations. It has been designed as a distributed multi-agent system, using agents to maintain and exploit the experiences of individual commanders as well as to transform suggested past plans into potential solutions for new problems. The system is mixed-initiative in the sense that a commander, through his or her agent, can view and modify the contents of the shared repository as needed. The agents interact through a common knowledge repository, represented by a blackboard in the initial architecture. The blackboard architecture is well suited for dealing with uncertain, complex situations that are part of modern warfare.

The fast pace of change and innovation in the software design community suggested that the DEEP program utilize a flexible and extensible architecture throughout the course of the effort to enable new ideas to be rapidly prototyped, evaluated, and then either discarded or used to evolve both the underlying architecture and the more operationally motivated functionality to which it is applied. This approach proved very effective as not all of the early goals were, in fact, realizable within the time and resource constraints of this effort. The severable nature of the components allowed the research team to make minor modifications to the system without resort to wholesale redesign and to rapidly investigate promising opportunities.

2.1. Problem Statement

The United States and other highly industrialized nations have developed military capabilities that excel in conventional force-on-force warfare, especially where tactics are well developed and understood. However, modern adversaries have devised the strategy of not going "head-to-head" with these capabilities and instead combat modern conventional forces with unconventional tactics, the aptly named *asymmetrical warfare,* e.g., the reliance on small arms and improvised explosive devices seen in the southwest Asian theatres of conflict. (Allen, 2004)

To meet these future challenges, United States forces have gone through a transformation where they not only support traditional high-tempo, large force-on-force engagements, but also smaller scale conflicts characterized by insurgency tactics and time-sensitive targets of opportunity. This transformation in turn has driven a need for a vastly new C2 process that is adaptable to any level of conflict, provides a full-spectrum joint warfighting capability, and can rapidly handle any level of complexity and uncertainty.

To support this end, the United States. Air Force (USAF) has championed a move towards a model of continuous air operations not bounded by the traditional 24-hour Air Tasking Order (ATO) cycle. Meeting these objectives will require a highly synchronized, distributed planning and replanning capability. As a potential way ahead, the Air Force Plans Office (AF/A5)

released in May 2006 a revolutionary vision paper titled "C2 Enabling Concepts" (Braun, 2006) depicting what a potential future C2 environment could be. Four key concepts emerged as being critical to the success of a future Air Operations Center (AOC):

- Distributed/reachback planning – "Today's constantly engaged AOCs no longer focus on shifting from one relatively rare major combat operation to the next. Their challenge is day-to-day, steady state C2 of these continual lower-end contingencies."
- Redundant/backup planning – "AOCs can be geographically separated but electronically plugged into the battlespace so that they function as if their members were co-located."
- Continuous planning – "The supporting and supported AOCs will maximize use of distributed network capabilities to ensure information is backed up and the supporting AOC is prepared to assume operations should the engaged AOC encounter a catastrophic event that makes operations unsupportable."
- Flexible, scalable, tailorable C2 – "...rapidly adapt to the level of conflict by connecting with worldwide capabilities, including joint and coalition forces."

Experience with recent operations also reveals that the C2 process must transition from a process of *observation and reaction* to one of *anticipation*. To that end, the focus of the research reported here is on developing a C2 environment that supports the vision of *Network Centric Operations* (NCO). (Alberts, 2007) The tenets of NCO are:

- Information sharing
- Shared situational awareness
- Knowledge of commander's intent

2.2. Initial DEEP Project Objective

In response to the need to support these key NCO tenets, the initial long-term goal of the Distributed Episodic Exploratory Planning (DEEP) project was to develop, in-house, a prototype system for distributed, mixed-initiative planning that improves decision making by applying analogical reasoning over an experience base. The two key functional objectives of DEEP were:

- Provide a mixed-initiative planning environment where human expertise is captured and developed, then adapted and provided by a machine to augment human intuition and creativity.
- Support distributed planners in multiple cooperating command centers to conduct distributed and collaborative planning.

That is, the architecture of DEEP was explicitly expected to support the key tenets of NCO in a true distributed manner. Because DEEP is not based on any current C2 system, the architecture was designed to:

- Combine planning and execution as a single integrated process of dynamic replanning
- Examine machine-mediated self-synchronization of distributed planners
- Experiment with the impact of trust in an NCO environment (e.g., "Good ideas are more important than their source").

Recommended high priority research topics (Alberts, 2007) to systematically explore included:

- Taxonomy for planning and plans;
- Quality metrics for planning and plans;
- Factors that influence planning quality;
- Factors that influence plan quality;
- Impact of planning and plan quality on operations;
- Methods and tools for planning; and
- Plan visualization

Although the DEEP project was able to deliver a stable and effective research platform by the middle of the project as planned, the above goals failed to take into account the availability of domain data (case bases), operational expertise or external interfaces to simulators, analytical tools, etc. (Distributed Planning in a Mixed-Initiative Environment Collaborative Technologies for Network centric Operations, 2008) This report describes our approach to achieving this vision of NCO and presents the progress to date on the development of the DEEP prototype, especially as it relates to these high-priority research topics. In short, the degree of research needed had been underestimated and in 2009 the scope of the project was adjusted to better match the timeframe and available resources.

2.3. Revised DEEP Objectives

The objectives for the second half of this effort were less operationally focused and more technology driven. The DEEP platform was used to explore inherent technical issues associated with the general blackboard/agent/case-based reasoning approach as well as develop rudimentary applications to identify the potential benefits, and hurdles, to transition of the technology to operational users.

In particular, a need was identified to refine the plan representation scheme, develop appropriate and relevant cases, merge and deconflict partial plans, and to develop protocols and processes for adapting, critiquing and modifying plans. Particular attention was directed toward investigating technical issues associated with:

- Defining methods for identifying past cases relevant to current events
- Adaptation of past plans (cases)
- Merging of plan fragments (cases)

- Adjudication of conflicting plan options
- Methods for critiquing plans.

Additional areas of interest include but are not limited to: knowledge elicitation and representation for the case-base and episodic memory, analogical reasoning for experience retrieval, and principled methods of case adaptation for creating skeleton plans.

Research objectives for the latter half of the effort were further distilled down to:

- Creation of representative cases by which to exercise the technology
- Ensuring plan representation adequately supports air, space, cyber and integrated battle planning needs, including a detailed plan representation schema and evaluation of schema merits
- Demonstrating coherence-based constraint satisfaction capability
- Demonstrating semantic-based case retrieval capability
- Developing plug-n-play integration capability for ad-hoc critic agents
- Demonstrating an integrated experiential planning capability

Emphasis on the distributed aspect of DEEP (Mission Assurance in a Distributed Environment, 2009) (Collaborating with Multiple Distributed Perspectives and Memories, 2009) (Mixed-Initiative Planning in a Distribued Case-Based Reasoning System, 2009) was deferred as was integration with a simulation engine for evaluating resulting plans. This is not to say that such capability is not supported by the DEEP design and architecture, only that limited resources did not allow for completion of those components of the larger DEEP vision.

3. Methods, Assumptions and Procedures

This chapter presents an overview of the DEEP research platform or architecture, a description of the components comprising the DEEP prototype, and a discussion of how these systems interact.

3.1. DEEP Research Platform Overview

The detailed design and working of the DEEP architecture have been documented in earlier reports. (Lachevet, et al., 2008) (Carozzoni, et al., 2008) (Synthesizing Disparate Experiences in Episodic Planning, 2008) As such, discussion will be limited in this report to an overview of that architecture.

Figure 1 depicts the DEEP system-of-systems architecture, comprised of the following Sub-systems:

- *Distributed Blackboard* for multi-agent, non-deterministic, opportunistic reasoning
- *Case-Based Reasoning* system to capture experiences (successes and/or failures)
- *Multi-Agent Critic System* for mixed initiative planning
- *Constructive Simulation* for exploration of possible future states

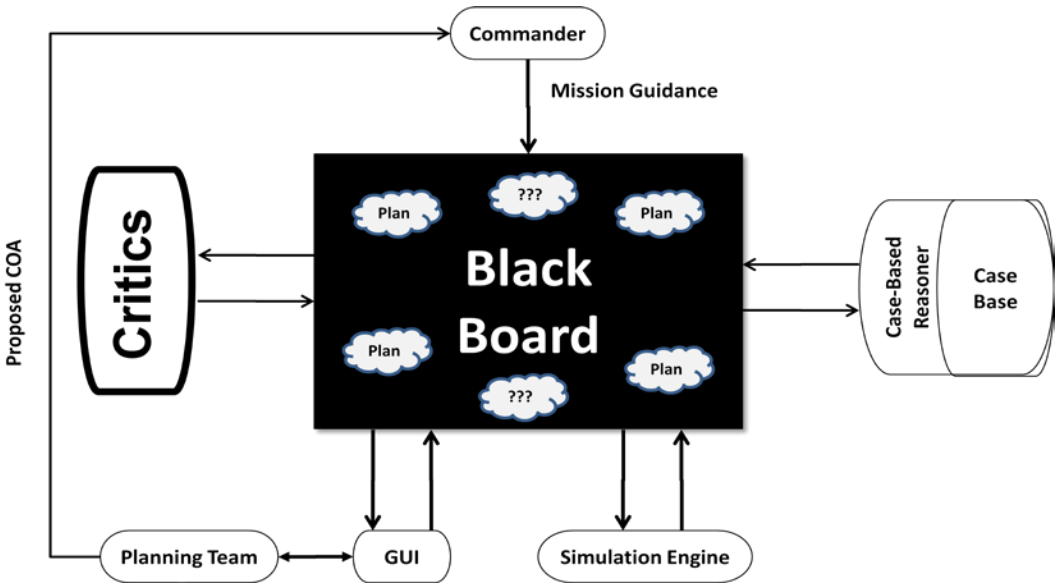

Figure 1: Notional DEEP Architecture

Integral to this design is the use of the ARPI Core Plan Representation as a common medium for human-to-machine dialog. As noted earlier, the Simulation Engine sub-system was not implemented, nor was significant effort applied to exploring and implementing the distributed blackboard aspect of the DEEP vision.

The DEEP architecture also includes a messaging system, various knowledge objects, a shared data storage system, along with a number of agents, all described later in this chapter. A typical dataflow between the pieces of this architecture is shown in Figure 2.

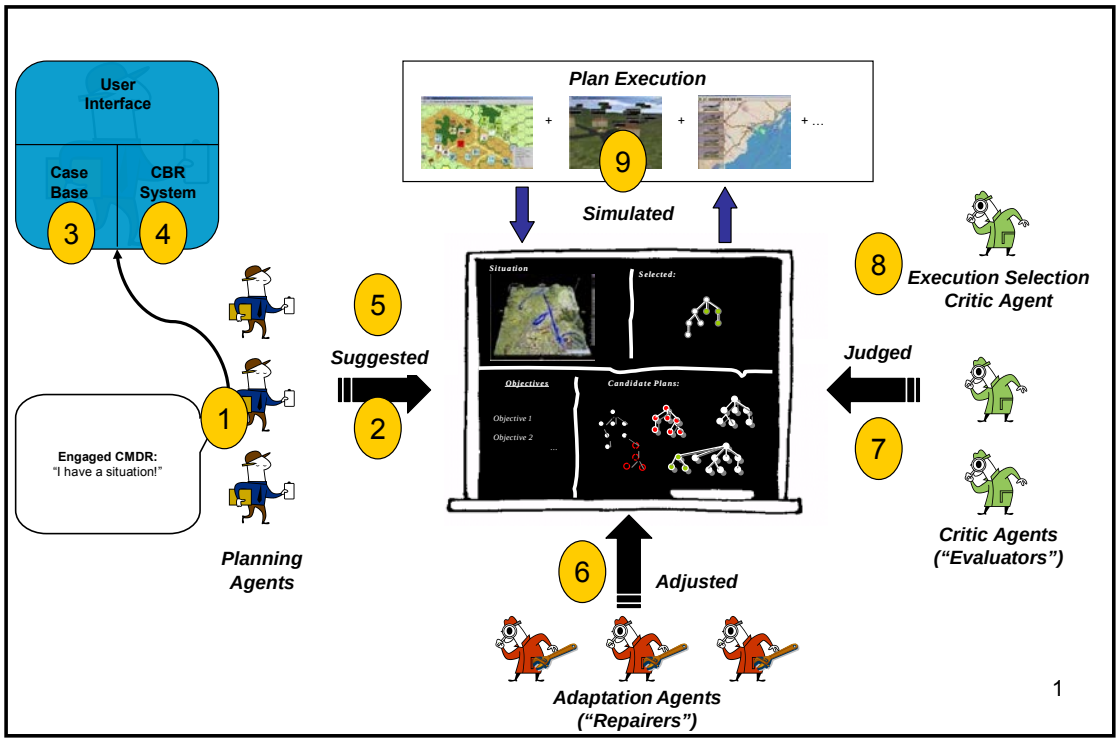

Figure 2: DEEP Dataflow

(1) This is the starting point for entry of data into the system, e.g., a commander or his staff may use a *planning agent* to describe a new mission.

(2) The planning agent allows for the commander to input information into the system which defines a set of *current objectives*. These objectives, along with other information, such as resources, locations, and time constraints, are collectively known as the *situation*. This situation is then placed on the shared blackboard.

(3) The blackboard would in turn notify all registered systems of the existence of the new situation. Using the given situation, the other planning agents, with their associated case bases and cased-based reasoning capabilities, would each search their case base for relevant past experiences.

(4) These results are then modified to fit the current situation.

(5) The results are posted to the blackboard as *candidate plans.*

(6) Once the candidate plans are on the blackboard, they are adapted by specialized *adaptation agents* to further refine these plans to meet the current situation.

(7) These plans are then critiqued by the *critic agents* which concurrently scrutinize the candidate plans and score them based on their individual expertise

(8) Once the plans are scored, the *execution selection critic* gathers the adapted plans along with their scores, determines their overall scores, and selects a number of top rated plans to be executed.
(9) The top rated plans are now executed in a simulated environment.

Once a plan completes execution, the results are combined with the plan and assimilated back into the original planning agent's case base.

Although this planning and execution is described in terms of a single flow through the system, in reality few plans will execute without changes. The DEEP architecture supports the modification of currently executing plans through feedback of partial results of plan execution into the blackboard. This allows the plans to be run through the adaptation and critique processes as many times as needed. One should bear in mind, however, that in this type of mixed-initiative system, there will rarely be a clean path from the initial planning problem to the final solution.

3.1.1. Blackboard Architecture

The key, real-time framework around which DEEP is constructed is that of a blackboard system. (Engelmore, et al., 1988) A blackboard system is an opportunistic artificial intelligence application based on the blackboard architectural software engineering paradigm. (Corkill, 1991) The blackboard system functions as a central knowledge store facilitating communication and interaction between the different software systems, including interface agents, critic agents, and simulation engines shown in Figure 2. These interactions are made possible by the sharing and passing of objects, be they requests for data or services, intermediate planning artifacts, plan fragments or assessments of plan components.

To meet the requirements of the initial DEEP vision for distributed C2, a distributed blackboard system was required. Current commercial and open source blackboard system implementations are not distributed, so the paradigm needed to be extended from a monolithic to a distributed environment. The current DEEP blackboard, shown in Figure 3, was designed and implemented following this extended view using previously described design patterns. (Hughes, et al., 2003)

A traditional blackboard system consists of three discrete components:

- Central repository of knowledge objects
- Knowledge Sources which are specialist software modules (agents in the case of DEEP) that provide specific expertise required by the system
- Control component which controls the flow of objects and problem-solving activity in the system. (Corkill, 1991)

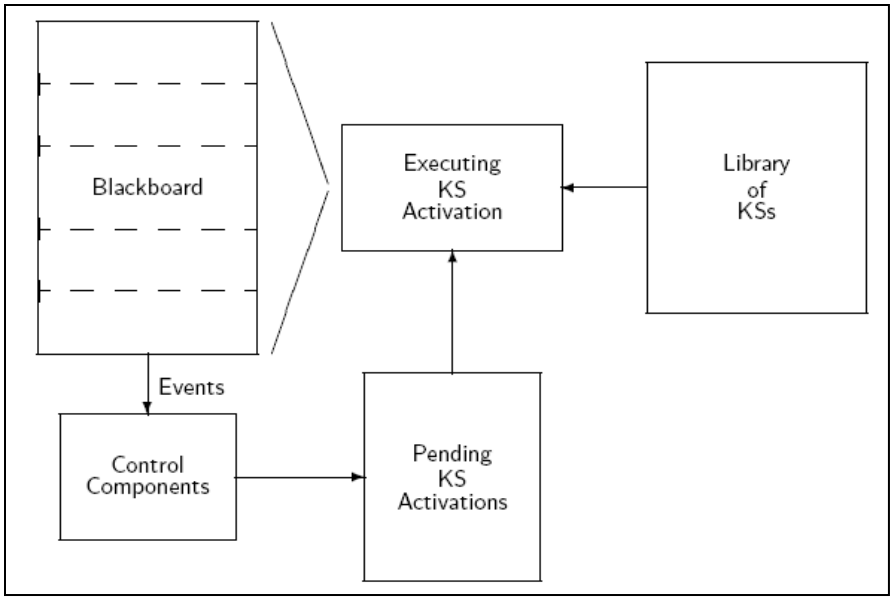

Figure 3: Basic Blackboard System

3.1.1.1. Blackboard Central Repository

The core knowledge structure of the blackboard is the global knowledge store labeled as "Blackboard" in Figure 3. This knowledge structure holds the objects within the system and is accessible by all of the system's knowledge sources. Because there could potentially be an extremely large number of objects placed on and contained within the blackboard at one time, blackboard data structures are conventionally divided in more than one way. These divisions are known as panes and layers, and could potentially contain further dimensions of separation. (Corkill, 1991)

In the DEEP architecture, the core knowledge structure is defined to provide certain functionality. The knowledge store component of the blackboard has been abstracted out to allow for future revisions and extensions to how and where the knowledge is stored. This interface allows the option for the backend of the blackboard to be replaced with a database or other high performance data store.

3.1.1.2. Blackboard Knowledge Sources

By connecting to the blackboard, an application has the ability to become a *knowledge source* of the blackboard system. Knowledge sources are independent agents that provide specialized expertise that contributes to the solution of a problem. A key characteristic of knowledge sources in a blackboard system is that they require no knowledge of the other knowledge sources that are connected to the blackboard. They bring their specialized expertise to the system and do not rely on others to provide it. Each knowledge source is responsible for knowing when, what,

and how it may contribute to the solution to the current problem on the blackboard. (Corkill, 1991) In the DEEP architecture, all systems must implement and conform to an interface provided by the blackboard in order to connect to it. The connection process includes connecting to the local blackboard proxy and registering with the blackboard for blackboard update event notifications.

3.1.1.3. Blackboard Control Structure

There are several control system paradigms that may be employed when designing a blackboard system. It may be very centralized to the blackboard, distributed among the blackboard and knowledge sources, or pushed out to the knowledge sources, requiring them to facilitate their own control of contributions to the problem. (Corkill, 1991) The blackboard system developed for the DEEP architecture splits the control between the blackboard application and its knowledge sources. The control component on the actual blackboard application directs communication among the distributed blackboards, where the knowledge sources are held responsible for choosing whether or not they should interact with new or updated objects on the blackboard, or even taking initiative and placing a new object on the blackboard without waiting for a blackboard event notification.

The control component of the blackboard is also what enables multiple blackboards to remain synchronized and distributed. Because the control component manages all of the activity occurring within the blackboard system, it is able to control how information is distributed among the connected blackboards, as well as maintaining synchronization through the use of queues and messaging schemes. This is what allows the blackboard system to be viewed as a single logical blackboard, while physically there are multiple, synchronized replicated blackboards. When a new object is passed to the blackboard proxy by a knowledge source, it is passed through the control mechanism, which distributes it to all connected blackboard applications. After all the connected blackboard systems receive the new object, they are placed in their local data store waiting to be manipulated or retrieved.

This aspect of the DEEP blackboard approach is only minimally developed. Although a rudimentary distributed implementation is supported, most recent applications of DEEP have treated the blackboard as a singular entity, focusing on issues of knowledge source responsibility, case-base retrieval and plan representation, and mixed-initiative interaction.

3.1.1.4. Additional Blackboard Components

In addition to the above, primary traditional blackboard systems components, the distributed blackboard designed for DEEP includes additional components that are necessary for a distributed blackboard and uniform communication between the knowledge sources connected to the distributed blackboard system.

A special interface, called a proxy, is provided by the blackboard that allows a knowledge source to connect and interface with it. This proxy connection is established using a network socket. Originally, a blackboard application was designed to be running on each computer that contains one or more knowledge sources. However, because each knowledge source connects using a network socket, it may reside on a separate computer. This proxy allows the interface to perform actions to the blackboard such as the posting and retrieval of objects. Other actions include the retrieval of an object by its unique identifier and the registration of new blackboard listeners. Similarly to the core data structure, the proxy interface could easily be extended to accommodate integration with other applications (new or existing) as needed.

A well-defined common interaction language is also necessary for a successful blackboard system. To keep the distributed blackboard as flexible as possible, the blackboard provides a simple interface to the knowledge sources for objects to be placed on the blackboard. This interface forces objects placed on the blackboard to contain certain properties and functions so the blackboard can work with the object. The properties include the partition the object belongs to, a unique identifier (UID) for each object, and a timestamp. By implementing this interface, the object also becomes serializable, allowing it to be transmitted over a network socket.

One of the main blackboard utilities is the *Packet*. A packet in this context is utilized by the blackboard control system to send messages to other connected blackboards, and is what the knowledge sources receive when they get an update event from the blackboard. Depending on the packet type, it contains certain useful information, some containing blackboard objects.

Another blackboard utility is the Blackboard Unique Identifier (BBUID), which is a unique identifier across a network. This UID is required for all blackboard objects, system, and knowledge sources. There are also other convenience utilities such as a log writer and properties file parser.

3.1.2. Core Plan Representation (CPR)

The various DEEP systems all use a common knowledge representation to facilitate their interactions. The future of military planning is broader in scope than just the Air Force, and will involve participants from various agencies (both military and civilian), possibly planning at different levels of abstraction. Thus, DEEP was designed to support plans for joint, coalition, and civilian operations as well handle plans at different abstraction levels (i.e., strategic, tactical, or operational). Planning for heterogeneous operations also means that the plan representation has to be able to consider the semantics of terms used in the plan, ensuring agreement among all participants. Finally, because DEEP is envisioned as supporting a mixed-initiative environment, the chosen plan representation must be easily machine-readable as well as intelligible to a human user.

In the 1990's the ARPA-Rome Laboratory Planning Initiative (ARPI) conducted research on several plan representations. CPR was the culmination of that work. Shown in Figure 4, CPR was selected for DEEP as best meeting the above criteria. CPR is an object-oriented structure that is agnostic to the planning abstraction level (i.e., strategic, tactical, or operational). (Pease, 1998) Its natural object-oriented structure also lines up very well with the machine reasoning capability that DEEP requires. The original CPR structure shown in Figure 4 was modified and extended to meet the needs of DEEP.

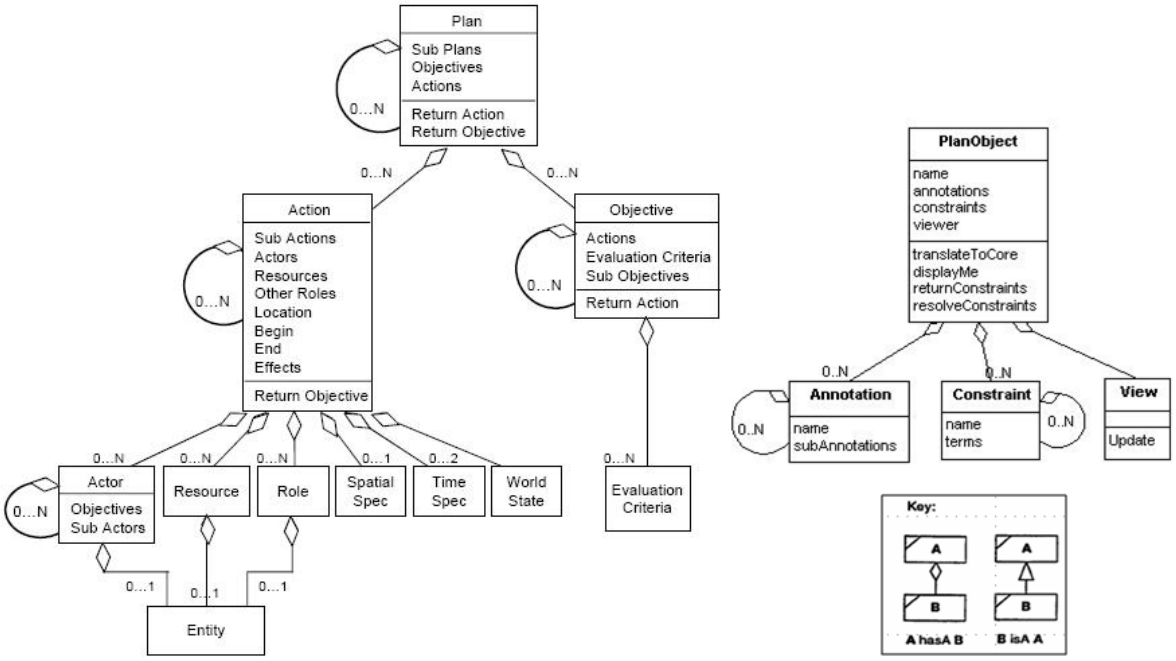

Figure 4: ARPI Core Plan Representation (CPR) Design (1998)

In DEEP, CPR is used to represent individual experiences, or *cases*, which are composed of a plan, events, and one or more outcomes. The attributes of the plan are used by the cased-based reasoning system to determine the similarity of past cases with the current situation.

CPR is the foundation for the DEEP architecture and used by all components. Because of this unanticipated use of CPR, the resulting DEEP-CPR has a number of significant changes from the original ARPI CPR in order to support a much deeper plan reasoning capability:

- *Annotation* (including *Fact* and *Assumption*), *Constraint*, *Imprecision* and *Uncertainty* were kept to support the larger need for representing uncertainty and the representation of relationships between objects, e.g., ($Object_1$ → $Object_2$) to allow more elegant statements in the future
- *Uncertainty* and *Imprecision* were deleted as they did not appear in any of the initial cases considered, and added complexity without any indication of benefit

- *Frame* aspect of *Annotation,* and *View* aspect of *Plan Object* were considered part of user interfaces and were deleted
- *Plan* and *Actio*n were kept as "holders" for other objects, i.e., detailed objects to represent who, what, when, where, why and how questions
- Retained *PlanObject* superclass to allow any object to have constraints, annotations, imprecision and uncertainty
- *World State* was removed as DEEP did not use "qualifier" objects
- *Evaluation Criteria* was removed as it lacked a common language with the effects of actions
- *Entity* was removed as it was not needed under the DEEP approach
- *Constraints* used as a "language for tying *Actions* and *Objectives*; however, *Actions* now needed *Preconditions* (which were missing entirely) and *Effects* (which were unspecified with the loss of *World State*). *Objectives* now needed *End States* to indicate when the *Objective* was met.
- *Objectives* specified by *Purpose*, *Method* and *End State*
- *Case*, a new object, was added to replace *World State* and provide a means to store information for use in case-based reasoning and case recall. Case has three parts:
    - *Plan* – What was intended to happen
    - *Event* – What went wrong, or happened not as intended
    - *Outcome* – What accomplishment or failure resulted from it? Initially broader, this was simplified to include only met and failed goals

The resulting DEEP-CPR representation for a *Case* is shown in Figure 5. Simplified views of the corresponding CPR and DEEP-CPR data structures are shown in Figure 6 and Figure 7 respectively.

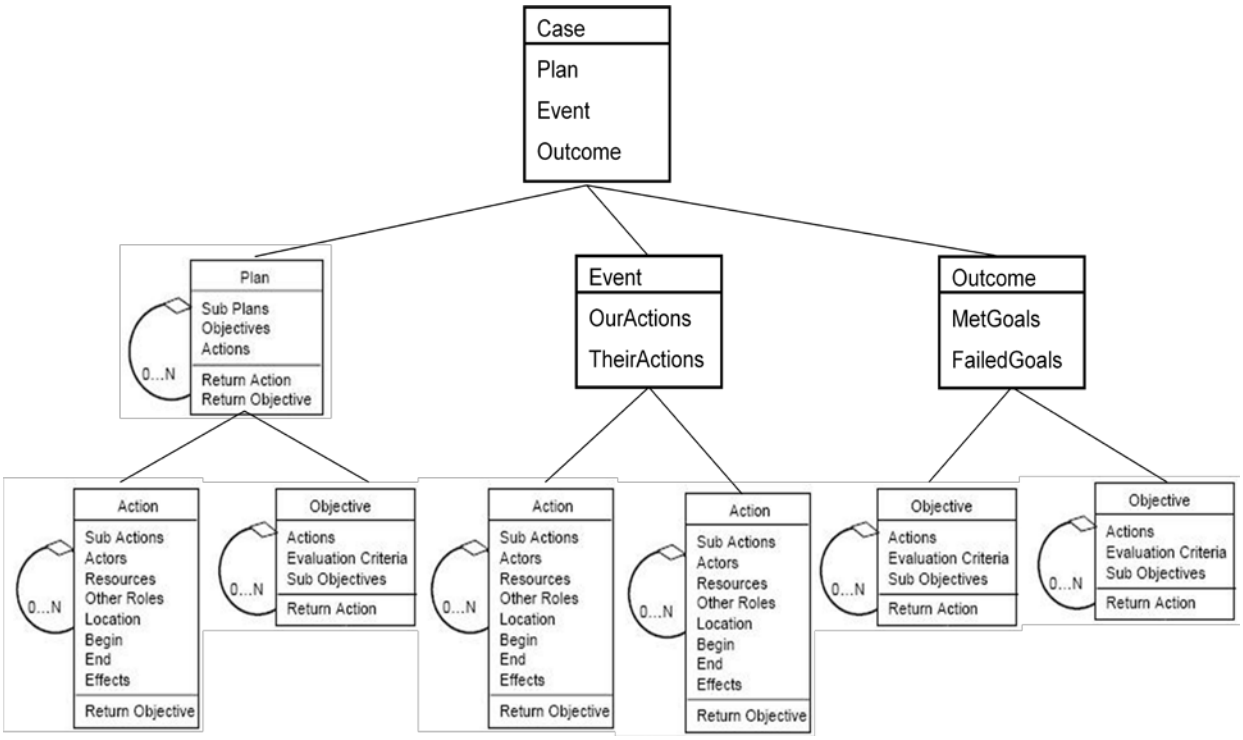

Figure 5: DEEP-CPR Representation Design

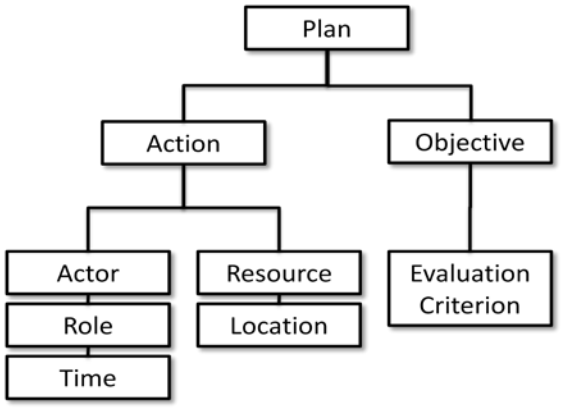

Figure 6: Simplified View of Original CPR Data Structure

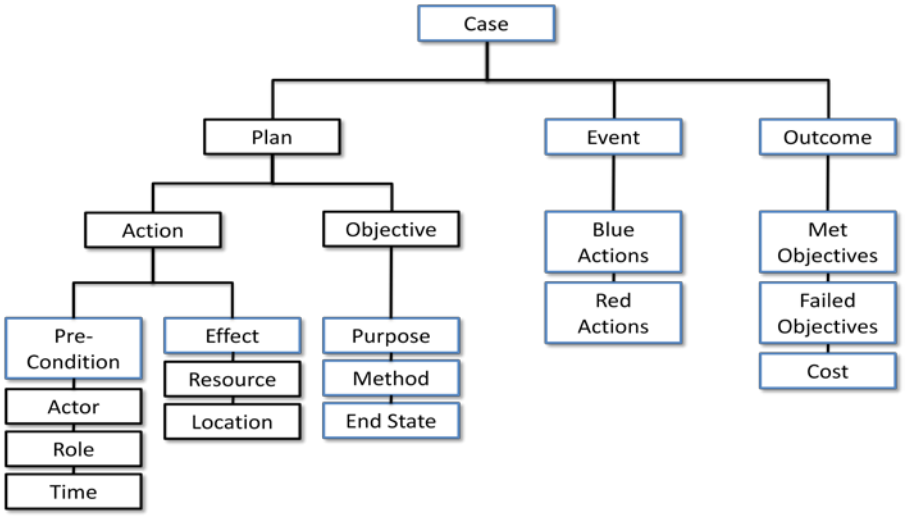

Figure 7: Simplified View of DEEP-CPR Data Structure

3.1.3. Case-Based Reasoning / Case-Based Planning

Case-based reasoning (CBR) (Kolodner, 1993) is an AI approach that has successfully been used in law, medicine, and computer help desk applications. Essentially, anything that involves utilizing solutions from past experience. In CBR, past experience is formed as discrete cases, with a clear beginning and end. This is different from episodic memory, in that episodic memory is often based on more loosely defined chunks of time. For successful CBR to occur, a primary enabling technology is the formation of sound analogies. While humans are adept at this form of intuitive reasoning, machines need to be similarly enabled through sound reasoning. Approaches to analogy forming include feature-based matching, semantic matching, structure-based matching, and multi-constraint theory.

Case-based planning makes use of past experiences to implement new plans and retain their outcomes. The planning agent uses a situation as an input to begin the process. The situation includes statements about the problem's objective, locations, actors, resources, and times. The case base for each planning agent may be unique.

Once a situation has been placed on the blackboard, the blackboard will broadcast a message notifying all registered systems about the new problem. The listeners in the other planning agents determine what type of object was placed on the blackboard, and react to a new situation by initiating cased-based reasoning for the new problem. The CBR process selects the best set of cases from its case base and posts them onto the blackboard as candidate plans. Once the candidate plans are placed on the blackboard, they are processed by the critic agents (discussed in detail in Section 3.1.4 and Section 3.3.3)

Each planning agent is expected to have a unique case base, since each planning agent represents the experience of some entity or group of entities. The case base of an entity can contain experiences of any kind. This variety is readily supported by DEEP's plan representation, CPR, because of its ability to work with planning knowledge at different levels of abstraction. Little processing is done by the planning agent itself, but rather by an external system that it interfaces with (e.g., CBR System). The agent itself is the medium between the reasoning process and the blackboard. (Creating and Capturing Expertise in Mixed-Initiative Planning, 2007)

3.1.4. Multi-agent Systems

Although multi-agent systems were identified as an important component of the initial DEEP vision, resources did not permit extensive research in this area. Related research was undertaken under the Mixed-Initiative COA Critics/Advisors research described in Section 3.3.3.

3.2. Revised Program Focus

Although the initial objective of this effort was to research, develop, and evaluate technologies for geographically dispersed commanders and senior staff to conduct intuitive, continuous, and distributed planning in the complexity, dynamics, and chaos of forth generation warfare, it became clear after a few years of research that a number of technical issues were beyond the timeframe and resources allocated to the effort. In 2009, the DEEP program was de-scoped to better concentrate on a few key research issues and less on the development of an operational planning capability.

These defining research challenges included:

- Refinement of the plan representation scheme
- Development of appropriate and relevant case bases
- Improving case retrieval (response time and quality)
- Merging and deconfliction of partial plans
- Use of software agents for adapting, critiquing and modifying plans

3.2.1. Case-Base Development

It became evident during the early years of this project that even the best blackboard framework, case-based retrieval process, and collection of agent-based plan modifiers would be dependent on a sizeable, robust and quality set of cases if it were to produce quality plans as output. Equally important is the need for such a case-base during the development process to ensure that the technology was effective and efficient outside of "toy" domains, and that the embedded algorithms and processes were not unduly biased by a small set of examples.

The early stages of the effort focused on crafting and encoding real-world examples, from naval/land/air engagements from World War II to contemporary humanitarian actions. Few if any of these missions are succinctly encoded, or had sufficiently available data to allow the team to do the encoding. It is easy to select the "best" case if there are only a hand-full of cases to pick from. Seeking to avoid biasing the developmental environment by hand-crafting specific cases, the team sought extant, operationally relevant and extensive case-bases to employ; thus avoid both the tedium, expense and pitfalls of doing it themselves.

In the end, two distinct domains presented themselves and have become the underlying domain(s) for the majority of the research in the latter half of the project: JAGUAR and StarCraft.

3.2.1.1. JAGUAR

The Joint Air Ground Unified Adaptive Replanning System (JAGUAR) was a DARPA program created in 2003 to develop technologies to enhance the capabilities of Air Operations Centers (AOCs). (Crumb, 2003) Among the programs tasks was the development of a capability to update models of assets and procedures that form the primitive elements of the plan. This will then allow a supervisor to quickly and accurately install new models into the overall JAGUAR software system.

The JAGUAR system was built as a model-driven planner to support a dynamically changing air mission planning and execution environment. The JAGUAR case base contains historical executed plans and objectives, along with the anomalies detected during execution of the underlying models, and plan message data (e.g., world state, monitored events), that was collected during the JAGUAR program.. Over the course of the program a large number of plans and missions were generated as the various tools were executed and their performance assessed. The data is stored as a repository of XML files.

In JAGUAR, the plan is a continuous entity that is comprised of one or more missions. For the JAGUAR domain, the data available (historical plans, world state, objectives) is present in a set of XML files. Hence the plan case base for JAGUAR is a case base of missions with information on how they were executed during a given run of the software. A trial can be viewed as a plan with a given time interval, e.g., all of the missions that were executed for a given day. The initial plan data and the world state were archived for 1000s of missions.

This domain is being used by extramural contractors, principally Raytheon BBN Systems, as they explore related anticipatory planning issues using the DEEP architecture and platform as the core infrastructure for their work.

3.2.1.2. StarCraft

StarCraft is a popular military science fiction force-on-force real-time strategy game developed by Blizzard Entertainment and was initially released in 1998. Game-play involves force-against-force engagements between red and blue forces which build an economy, construct physical bases and develop certain military capabilities to employ against each other. A high-level aspect of the game is the tradeoff between developing the economy and developing the army. If one side focuses too much on building the economy, they are susceptible to attacks from an opponent's army. However, the benefit of building a strong economy is to improve technology and build a more formidable army during the later stages of a game. There are many other important aspects to the game which help a player achieve victory, such as effective scouting of the opponent, setting up proper defenses, and countering your opponent's capabilities and actions. A screenshot of a typical StarCraft game in session is show in Figure 8.

Figure 8: StarCraft Screenshot

3.2.1.2.1. StarCraft Overview

Although StarCraft is a synthetic environment, is has a number of characteristics that make it an excellent domain for case-based planning research. We selected StarCraft base on the following aspects:

- Large case base – one of the most important requirements of a domain for experience based reasoning is the ability to obtain a large corpus of experience (cases) over which to reason. StarCraft has become very popular in the gaming world to the extent that it has a

large following and that success has generated international tournaments. The result of each of the games in these tournaments, i.e., a replay, has been made recorded and is available to the gaming and StarCraft community. Tactics can be extracted from these replays either by direct examination or by re-running them in the StarCraft game software.

- Complexity – although the StarCraft simulation engine is deterministic, the large action space over which a game proceeds makes it unlikely to see the same outcomes given a small set of actions taking place over a limited portion of a game's duration.
- Active research domain – this domain is actively being researched by a number of academic institutions and has been used as a research domain for several United States Navy sponsored contracts. (Defeating Novel Opponents in a Real-Time Strategy Game, 2005) (Aha, et al., 2006)
- Abstract C2 domain – this domain allows planning and coordination of multiple entities throughout the full spectrum from tactical to strategic military operations. Our work focuses on strategic level planning and lets previously developed heuristics-based components handle the tactical level details.
- Other Benefits:
    - Asymmetric forces (different object classes, planning and coordinating differing objectives)
    - Uncertainty
    - Quantitative metrics
    - Real-time Execution
    - Rich feature set
    - Interface for human Interaction

In 2008 a public domain application programming interface (API), called Brood War API (BWAPI), was developed within the StarCraft community using C++. It allows an application to control unit movement and extract the game state from StarCraft. This is achieved by directly manipulating the game engine while it is executing. Since the release of BWAPI, other developers have added additional functionality and implemented the API using other languages. Still other developers have created game controlling code, called "bots", for use in research and competitions. The result of all of this activity is the availability of thousands of past confrontations stored in machine-readable form and executable (via replay) in minutes. In addition the ability to evaluate modified game-plays, essentially courses of action, is invaluable in assessing the output of the DEEP planning process - and thus the efficacy of the DEEP technology.

A key component of this research has been the integration of the StarCraft execution engine with the DEEP research platform. This has required the development of additional software to:

- *Provide the ability to change strategies during execution*. Software on the DEEP side analyzes the current situation and determines when to re-plan. If it is determined that re-planning is called for during execution, this feature will allow DEEP to submit a new plan, thus enabling continuous planning. Since the DEEP platform is currently developed in Java and the BWAPI is developed in C++, each side must be able to interpret the action of the other software.

- *Determine when and how new software will be incorporated into the DEEP development environment*. As new versions of the BWAPI are released, they must be assessed and a determination must be made as to whether the release should be incorporated into the working version of DEEP.

This integration of the StarCraft AP with the DEEP infrastructure allows for the automated creation of StarCraft units, rather than requiring the manual creation of prerequisite components. This feature greatly facilitated controlling the StarCraft engine from a tactical level, but was found to be inadequate for higher level strategic control. Various APIs were developed as extensions to the BWAPI API and were evaluated. After searching for an open source API that would provide the required functionality, BTHAI, a StarCraft "bot" which utilizes the BWAPI, was determined to best meet the requirements of the DEEP development effort.

BTHAI extends the core BWAPI, providing an agent-based architecture for controlling each type of unit. The BTHAI source code was examined and found to be well formatted and documented, and logically organized. BTHAI's use of separate files to control the operation of the API: one for build order, one for technology upgrades, and one for squads, lent itself to serving as a basis for the representation of a plan. The integration of BTHAI with DEEP is continuing under DTIC contract FA1500-10-D-0005/0003, "Anticipatory Planning Integration and Demonstration".

Although the DEEP project has ended, the DEEP framework remains an important research tool within the Information Systems Division of AFRL and its support and management is being continued under other projects. StarCraft has proven to be an excellent domain for exploration due to its optimal complexity and large set of encoded cases.

3.2.1.2.2. StarCraft Case Base Definition

Once the DEEP team settled on StarCraft as a domain to use for research, a case base had to be developed to support the case-based reasoning process. These steps included designing case structure, selecting features, formal logic for a StarCraft plan, and automation logic to convert StarCraft replays into cases.

As described in the StarCraft Overview, Section 4.2.4, one benefit to utilizing StarCraft as a data domain was the large amount of publicly available replays available to us. Using built in game functionality to save replays, it is easy to grow the case base over time. A replay is a saved game

that can be viewed later on or even executed again to monitor what happened. Combining an open source API for parsing replays with a developed programmatic option to automate reading the replays and parsing out the required information, the DEEP team was able to quickly generate thousands of cases.

Before creating the case base, the DEEP team conducted internal discussions using empirical evidence of game play to support a selection of features that would support initial case construction. It was also decided the cases to be used would be one player versus one player matches, so no games with more than two players are implemented in our platform. The importance of what features to use was an iterative process. As the project continued the team reviewed the work of other teams that were developing their own AI systems, such as bots, to be used at the StarCraft AI conference such as the Berkeley Overmind Project (11UC) to see what other features were being created by the AI community for possible inclusion into the DEEP system.

We used a small set of features (attributes) to describe a particular case (game). Figure 9 shows the features used to describe a case. We used a time-step of two-thousand game frames, so for some of our features we are measuring values every two-thousand frames. These features are as follows:

The state lattice is based on the build tree (or tech tree expansion) of the StarCraft (Broodwars Expansion Pack). Specific buildings enable new features and capacities like: building new types of combat units and buildings, researching new capabilities, etc. The SCReplay (11ht), an open-source java based project was utilized to read contents of saved StarCraft games. This approach enabled us to effectively reduce the complexity of representing an entire game state through segmentation. Feature data for an example case is shown in Figure 10.

Game Features	Player Features (for each player)
Unique Identifier	Player name
Replay name	Player race
Map name	Count of buildings built
Winning player name	Count of units built
Winning player final state lattice	Count of workers built
Actions per minute of game	Expansions and their game frame
	Final player state lattice
	Vector of player states by time step
	Unit type and production count at each time step

Figure 9: StarCraft Case Base Feature Set

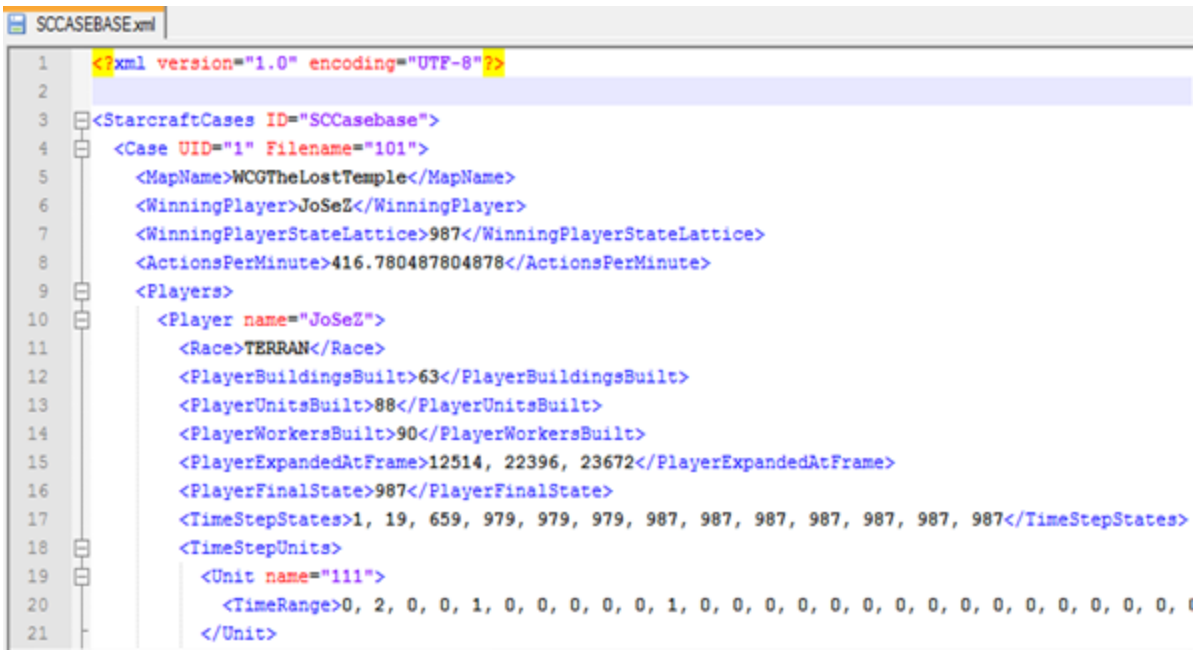

Figure 10: Sample StarCraft Case with Feature Data

Most of the information in a case is used by retrieval to obtain the best match to a given situation; however there is part of the case structure that is utilized later on by the plan execution agent to implement the plan. All the information under the units at each time step describe what was built and at what time. This is the build order the player used in the game and consequently, is the action list the plan execution agent will use to construct buildings and build units. The data for player expansions is also used to allow the plan execution agent to know when to build an expansion.

Initially, one case base was created and formatted in an XML file. Once the move was made to have multiple retrieval agents, one for each Terran versus race pair in the game to mimic having multiple planning point of views or planning centers, the single case base was converted to one overall case base containing all cases that can be used when there is more uncertainty in information, such as not knowing the race you are playing, and three other smaller case bases for each Terran versus race pair in the game that are used when the race of the enemy is known. There are more possible case bases, but since the DEEP system plans as a Terran player, there was no need to utilize the other case bases.

The DEEP team believes feature selection for this project is not complete and requires future work to support better plan generation, more specific player information, more specific mapping information, among others.

3.2.2. Information Retrieval

One of the key requirements of a case-based reasoning system, such as is envisioned in DEEP, is the ability to quickly and efficiently retrieve a right-sized set of relevant cases from a potentially vast set of stored entries. Critical to this is the proper storage of historical data as a case-base is assembled. Not enough distinction between cases can result in extraneous cases being returned and a case-base that is unwieldy in size for the amount of useful information it contains. Too much distinction can mean that the desired return case lies somewhere between two existing entries, but not close enough to either one to trigger a partial match.

3.2.2.1. Memoire

Upon realizing that the initial retrieval algorithm used by DEEP (Carozzoni, et al., 2008) would not scale well enough for multiple retrieval iterations as required by practical applications, the DEEP team conducted a thorough literature search on the state of the art for information retrieval techniques in case-based reasoning and similar domains. The team decided to implement a baseline information retrieval algorithm, known as "Memoire" (Bichindaritz, 2006), that utilized indexing techniques promising near-linear retrieval times. The resulting algorithm has two main parts: the indexing algorithm and the retrieval algorithm.

The indexing algorithm iterates through each case in the case base. Each one of these cases is analyzed by looking at each feature and indexing the value of that particular feature. The actual index is a list of values which have occurred with the index of the cases they have occurred in. As shown in Figure 11, for example, a single line in the index for a specific feature might be Value1 = 1,3 indicating that for this feature, Value1 occurs in cases with id 1 and 3. This example only shows one feature and there would be as many indices as features identified to be used in the particular reasoning domain. The result is a set of inverted indices for which there is a separate index for each feature. The decision to create a separate index for each feature was a diversion from the original Memoire algorithm and was done so for a number of reasons. The first reason is the value of the feature would have context within the feature of which it occurred. In the simplest example, the value "22" might have a very different meaning for the feature of temperature compared to representing a count for another feature. This context improved the accuracy of the retrieval algorithm. Also, the division of these indices slightly lowers the computational complexity of the algorithm.

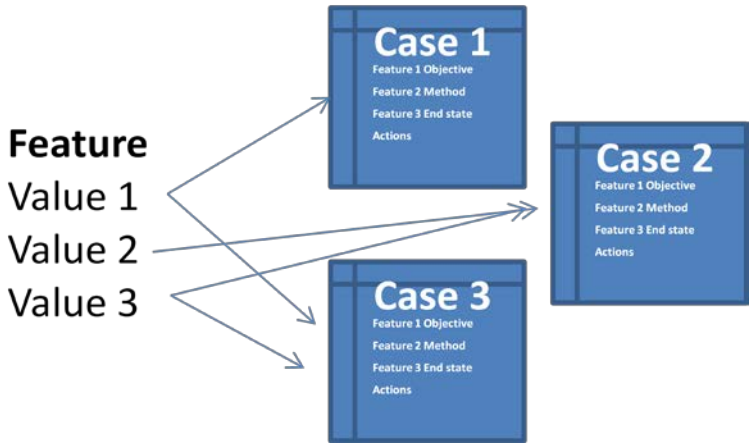

Figure 11: Inverted Indexing of Case features

Once the preprocessing of the inverted indices is completed and loaded into memory, retrieval of relevant cases is very efficient as all that is needed is to perform a simple lookup based on the case similarity score calculations. Figure 12 represents how a case's score is calculated. The algorithm begins with a set of features that it uses to match against the indexed feature sets. The algorithm iterates over the vector of features, and determines what cases match. This particular implementation uses an exact "Categorical Match" to increment a "hit score" for each of the cases for which a match is found. This process is extremely fast because it is using hash table lookups and the summing of occurrences. Upon completion of the iterations, a ranked order list of the cases which best match the input feature vector is created.

$$\text{Case score} = \sum_{\text{Feature 1}}^{\text{Feature n}} \text{Similarity Score (Categorical Matches)}$$

Figure 12: Similarity Score Formula

The implementation of the algorithm provided exactly what was needed: reduced computational complexity while retaining retrieval accuracy. One experiment measured the retrieval time between the old algorithm and new algorithm using a simulated case base that was developed for such experimentation. The simulated case base created thousands of cases choosing features from a random distribution of values from our taxonomies. The experiment used case base sizes of 100, 500, and 1000. The results showed the old algorithm with a linear increase in time with respect to the number of cases, with the time approaching three minutes with the 1000 sized case base. The new algorithm remained near-constant with respect to the number of cases and had a

retrieval time of just 390 milliseconds with the 1000 case case-base. The results of this experiment can be seen Figure 13 below.

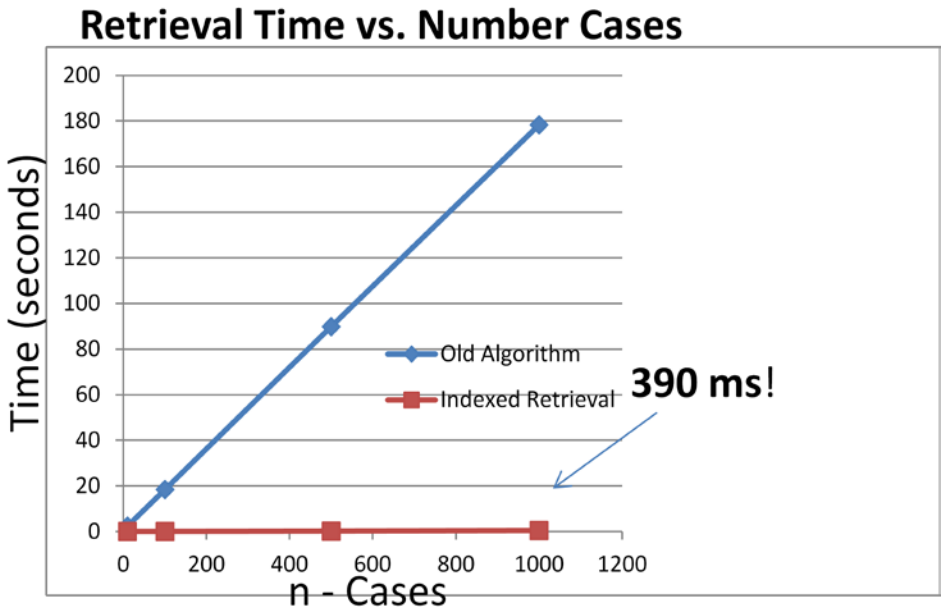

Figure 13: Indexing Algorithm Effectiveness

It must also be noted that that even though the DEEP team did not experiment with the computational complexity with respect to feature size, it is expected that the complexity was reduced from exponential to linear.

With this approach there are several tradeoffs. The first tradeoff was removing of the run-time similarity score calculation. Although the implementation is currently using categorical matches for the similarity function, this function is easily swapped out for a more complex similarity calculation function. This function resides in the preprocessing algorithm, therefore retaining the fast retrieval calculation times. Also, the memory usage is a consideration when using indexed retrieval. There is a cost to consider of having these indices pre-loaded into memory. However, with simple tables containing values and numbers representing case identifiers, it showed the memory usage to be extremely minimal.

3.2.2.2. Latent Semantic Analysis (LSA)

The DEEP team faced a challenge in preparing historical StarCraft log files for storage in a DEEP-friendly case-base. Latent semantic analysis (LSA) is a technique for analyzing documents based on both the semantic distance between words in the document and the proximity of those words as they are found in the document. (Using Latent Semantic Analysis to Improve Information Retrieval, 1988) The challenge for DEEP was to convert a continuous

StarCraft replay file into a set of discrete individual states based on the build lattice tree (for each race, respectively) that would then support efficient recall of cases. (Semantic Interoperability in Distributed Planning, 2008) (Semantic Adaptation AGEnts (SAAGE), 2009) (Improving Plan Adaptation Process Through Semantic Technologies, 2009)

A simple partitioning mechanism was devised to reduce the quadratic complexity of the clustering algorithm while preserving the quality of the search results. The singular value decomposition (SVD) is the most computationally expensive step of the LSA algorithm used to align documents based on their latent semantic proximity with respect to neighboring documents. The new approach is based on the premise that only the top set of search results are important and that performance can be improved by loosening constraints on the proper rank order of search results. The claim is that, in a majority of searches, users are interested only in the set of top ***k*** results. For example, according to one study (Eye-Tracking Analysis of User Behavior in WWW Search, 2004) the first five rank positions in Google search results get over 88% of traffic, of which the first three positions get most (roughly 79%) of the traffic. Furthermore, roughly 96% of users never left the first page of results, further reaffirming the premise that only the first top set of results are the most significant. With this in mind, performance should be significantly improved if the proper search order is only enforced for the top ***k*** results.

The resulting elementary optimization algorithm, ELOP, reduces computational complexity while still preserving proper rank order of top ***k*** results. The approach is to split the entire case-base into a set of ***p*** independent partitions. The SVD of each partition is roughly of the same size and still relatively fast to compute and within linear time. During the search step, the top ***k*** results from each partition are used to form a new term/document matrix. Next, the SVD computation is performed (at runtime) on the newly created term/document matrix. The computational complexity of the final SVD is significantly reduced as compared to the original data set. While this approach is not exotic, it does preserve proper ranking order for the majority of the top ***k*** results at the expanse of having loosely sorted lower ranking documents as indicated by the preliminary results.

Results from this work are still being analyzed and will be documented in a pending paper. (ELOP-DEEP, 2011)

3.3. Related Explorations

A number of related technical investigations took place over the course of the DEEP project. Some of these were directly related to the core aspects of DEEP, others sought to leverage the new capabilities provided by the DEEP platform.

3.3.1. Robust Coherence

The vision of DEEP includes using the collective experience of many individuals, stored in the form of distributed case-bases. Ability to semantically reason over this collective can assist commanders in their development of military plans. One of the technical problems posed by this approach is the need to differentiate between, and resolve contradictions between, retrieved relevant past experiences that, taken independently, give an appearance of a valid satisfaction of the posed mission need, but which taken together, contract each other or suggest an incompatibility or deficiency in the retrieved case.

3.3.1.1. Robust Coherence Overivew

In case-based reasoning, previous solutions are recalled and adapted to fit new problems. However, for complex problems with multiple stakeholders, multiple sources of experience should be considered to increase the diversity and effectiveness of such solutions. The DEEP team developed an approach for this called *robust coherence*. This approach combines two seemingly contradictory theories from the philosophy of knowledge: coherence and falsification. Using these two theories in concert, *robust coherence* seeks to justify contributions from several agents in a collective context that also corresponds to outside evidence. Using this approach, multiple agents can suggest actions and goals from experience to address a problem, and develop the best option based on the robust coherence among these experiences.

In the philosophy of knowledge, coherence refers to a theory about knowledge where coherent beliefs are beliefs that are mutually supportive in an overall context of justification. Coherent beliefs exist in a web, where complex interrelationships hold the beliefs together in a justificatory 'package' that makes up knowledge. (The Coherence Theory of Empirical Knowledge, 1976)

Coherence has been described as a set of constraints between elements in a system of beliefs. (Coherence as Constraint Satisfaction, 1998) Satisfying those constraints establishes the coherence of the system. This is accomplished by choosing to accept or reject beliefs based on how the various elements in the system 'fit together'. This approach requires the establishment of both positive and negative constraints based on how the elements in the system are related to each other. Certain relations between beliefs can be characterized as coherent (such as explains, associates, or facilitates), while others denote incoherence (such as incompatible, contradictory, or inconsistent). Elements that are related by a coherent relation are positively constrained

(should be both accepted or both rejected), while elements related by an incoherent relation are negatively constrained (one should be accepted while the other is rejected). Each of these constraints has a strength value, which indicates how much that constraint contributes to the overall coherence of the system. This allows us to employ soft-constraint optimization techniques on large packages of beliefs to determine the most optimal set of beliefs to accept. (MiniMaxSat: A New Weighted Max-SAT Solver, 2008) Coherence has been used as an approach to determine the appropriate actions and goals in a situation, an approach called *deliberative coherence*. (Thagard, 1995) The goal of this approach is to formulate a plan, consisting of actions and goals based on a system of constraints. This system is populated with facilitation and incompatibility relations that allow the set of possible actions and goals to be pruned using coherence.

We used this notion of coherence to choose a set of actions and goals *from experience* that will be acceptable to use in planning. Each agent suggests different actions and goals to address a problem, and the agents use coherence to formulate the best approach in the collective endeavor. In this way, suggested experiences exist in an overall context established by the whole group of agents. Expertise is exchanged collectively, leading to shared understanding of a problem.

While this form of coherence may appear to be useful to establish the justification of different options in a collective planning endeavor, there is the danger of forming a coherent set of beliefs that does not correspond with outside reality. Generally, this is called the *idealism* objection to coherence. In deliberative coherence, the acceptability of factual beliefs influences the relations in the system. (Thagard, 1995) In the following section, we will address the idealism concern by applying critical rationalism as a framework for influencing coherence relations. This will allow a group of agents exchange actions and goals from experience that correspond with evidence from the world the agents are trying to influence.

3.3.1.2. Critical Rationalism

To address the problem of idealism in coherence, the team turned to another form of reasoning to help inform coherence as an epistemology: a deductive view of truth that seeks to refute theories based on inconsistency with evidence. This method of reasoning is called *critical rationalism*. (Popper, 1963)

Under critical rationalism, theories are postulated and stood to the test of falsification. In other words, theories are considered which are potentially falsifiable, and then compared to a set of observations. If the theory holds up to this set, compared to competing theories, it is considered the least untrue (rather than most true). The measure of this aspect of *truth* is known as *verisimilitude*. (Popper, 1963) Logically, critical rationalism is based on deduction, rather than induction. This means that verisimilitude measures the degree to which a theory is able to stand

up to criticism based on what it deduces should be true. Whether or not those deductions match evidence tells us which theories are more reasonable than competing interpretations.

We can use verisimilitude to better inform a coherent set of experiences by attempting to locate information that refutes some of the aspects of the experiences. By doing this, we establish the degree of falsehood in those experiences for facing a current problem, and avoid the pitfall of blindly applying experience. This is what makes *robust coherence* different from ordinary coherence. Rather than relying on coherence as the only mechanism of justification for beliefs, robust coherence uses falsification to establish the *anti-justification* of beliefs.

The team incorporated falsification into the constraint-satisfaction interpretation of coherence in two ways. First, we provided our constraint-optimizing algorithm with an initial assignment of *rejected* to actions or goals which were shown to be false based on observations. Second, constraints involving falsified elements were amplified in strength to ensure that the optimization of the coherence system would more likely take into account the fact that some beliefs were initially rejected. In other words, by initially assigning *rejection* to falsified beliefs, and making their constraints to other beliefs more drastic in strength, the team could influence a constraint-optimizer to act in such a way that generated coherent beliefs which also corresponded with outside evidence.

Our approach uses coherence informed by critical rationality to create set of coherent, robust experiences which address a specific problem. Falsifying information allows us to examine how those experiences' utility is inhibited by facets of the ever-changing world. In this way, critical rationalism can also indicate critical conditions in the world, allowing for the discovery of new goals. In the following section, we will examine the mechanisms the team employed to accomplish a system of robust coherence for the task of planning.

3.3.1.3. Planning with Robust Coherence

In this section we will introduce a case-based planning approach that applies robust coherence to planning. As different planning agents (Robust Coherence: An Approach to Multi-Agent Experience-Based Planning, 2009) suggest experiences from their own case bases, a coherence agent interprets the actions and goals from these experiences in a system of deliberative coherence, while critic agents use information from the world to adjust these relations (reflecting falsification). This reasoning establishes the positive and negative constraints between these portions of experience, allowing a collective set of actions and goals to emerge as justified. This collective set can then be adapted and de-conflicted as a cohesive plan. Below we outline the guiding principles the team used to establish the facilitation and incompatibility relations between actions and goals in this approach:

- Let *C* represent a Constraint Satisfaction Problem that describes how different actions and goals relate to one another under the following conditions.

- M1. *Structure*. Actions stored in a case facilitate the successful goals of that case, and are incompatible with the failed goals of that case. Upon instantiation, the degree of similarity between the past and present goals weighs upon the strength of this relation.

- M2. *Effect Transitivity*. Information about effects and how actions achieve or avoid them indicate coherence relationships to other actions or goals.

    - If an action, *A*, achieves an effect, and another action or goal, *B*, requires that effect, then *A facilitates B*.

    - If an action, *A*, avoids an effect, and another action or goal, *B*, requires that effect, then *A is incompatible with B*.

- M3. *Competition*. If two actions must compete for resources in order to be accomplished, then those actions are incompatible with one another.

- M4. *Pragmatism*. If some goals are considered critical to success, then the initial assignment of *C* includes accepting these required goals.

- M5. *Falsification*. If an action or goal, *A*, includes preconditions that are not present in the outside environment, and no action under consideration can create those preconditions, then *A* is assigned as rejected in the initial assignment of *C*. Relations that involve *A* within *C* are amplified to a

3.3.1.4. Weighted Constraint Satisfaction

The robust coherence approach described in 3.3.2.1 above results in a constraint satisfaction problem (CSP) where a weight is associated with each constraint representing the amount of coherence that would be gained if that constraint were to be satisfied, thus known as a weighted CSP (WCSP). The issue remains, however, of how that weight is to be selected. The CSP solver itself must not be arbitrarily selected. The default implementation in DEEP was to use a static weight assignment. (Creating and Capturing Expertise in Mixed-Initiative Planning, 2007)

Alternatively, an investigation was undertaken to develop a dynamic weighted CSP (DWCSP) allowing for the dynamic addition and removal of constraints from the WCSP. (Hasseler, 2010) As the set of experiences (cases) available to the DEEP reasoned change, so too can the WCSP be updated in real time without needing to restart the WCSP solver.

The DWCSP algorithm and solvers were evaluated against the MAX-SAT canonical CSP benchmarks through the addition and removal of new constraints. (Argelich, et al.) Both the dynamic DMaxWalkSAT and real-time dynamic RDMaxWalkSAT solvers performed better that MaxWalkSAT on DWCSP problems. These algorithms are small and non-complex, which should encourage their implementation and inclusion in future systems.

We've presented the notion of *robust coherence*; a method of group decision making that seeks a maximally coherent decision while corresponding to the dynamic world. By combining seemingly contradictory theories of knowledge, robust coherence allows for group decision-making focused on both consensus and fact. We've also applied this approach to a case based planning algorithm called rob-coh, and implemented that algorithm to provide a test-bed for further research in coherence.

While our initial implementation focused on multi-agent planning, the basic approach of robust coherence can be applied to a wide variety of knowledge-based domains. For example, a robust coherence system can be established to discover what theories and observations a collection of agents hold in common. Using this coherent picture of a domain, the agents could use the selected theories to generate predictive data, adding to the set of observations and further refining the coherent theories.

3.3.2. Model Adaptation Using Software Agents and a Case Base

Initially separate from the DEEP program itself, a group of researchers with AFRL/RI were exploring the use of software agents to automatically adapt models, determining if and when supervision or suggestion by a human was appropriate. The models of interest were associated with the DARPA Joint Air Ground Unified Adaptive Replanning System (JAGUAR) program. This group teamed with the DEEP researchers to investigate using the DEEP blackboard framework in support of the former team's goals. Several of the JAGUAR tools were modified to operate within the DEEP framework and a number of software agents were developed to support autonomous model change. This work was conducted under the extramural research contract FA8750-10-C-0149, "Agent Based Model Diagnosis and Repair." (Mulvehill, 2011)

The final technical report for this work documents the results of this work, describing how the JAGUAR tools were extended to operate within DEEP, how software agents running on the DEEP blackboard were developed to support autonomous model change, and the results of the experimentation that followed.

The JAGUAR system was built as a model-driven planner to support a dynamically changing air mission planning and execution environment. The JAGUAR case base contains historical executed plans and objectives, along with the anomalies detected during execution of the underlying models, and plan message data (e.g., world state, monitored events), that was collected during the JAGUAR program.. A number of JAGUAR tools were available to provide

mixed-initiative support to the operator on model diagnosis and repair. (Expectation Failure as a Basis for Agent-Based Model Diagnosis and Mixed-Initiative Model Adaptation During Anomalous Plan Execution, 2007) In the DEEP application, specialized software agents that leverage several of the JAGUAR tools originally developed to support human operators were developed. The DEEP blackboard was used to support communication between all of the major elements. The final DEEP-Agents architecture is shown in Figure 14.

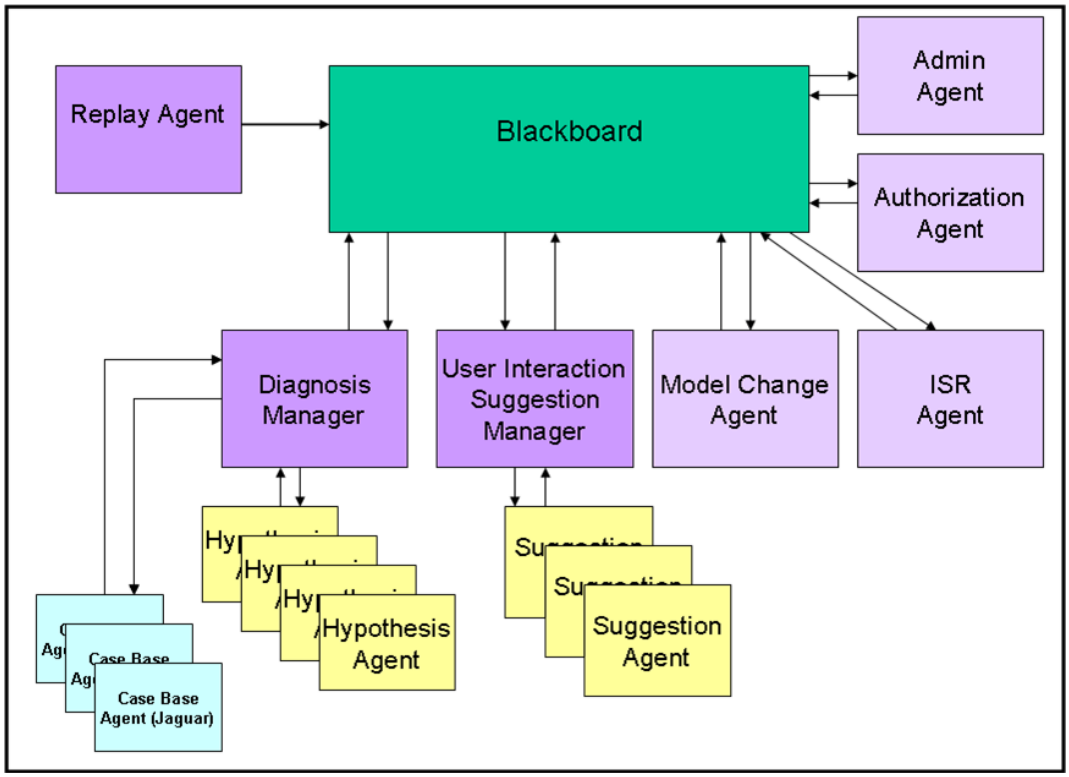

Figure 14: DEEP Agents Architecture

The use of DEEP for this effort produced two particular results of importance to DEEP: (1) Validation of the applicability and usefulness of the DEEP architecture and exercising of all key DEEP components in what was, in effect, a blind test as the JAGUAR program was not considered during any of the DEEP development stages; and (2) Using agents (via the DEEP blackboard) to modify models entities extracted from case bases is analogous to using planning agents to critique and modify plans and plan fragments extracted from other case bases. This latter result played a prominent role in the subsequent development of software agents, operating in a mixed-initiative environment, to construct viable courses of action (COAs) from partial plans retrieved from a case base(s) of past activities as described in Section 3.3.3 of this report.

3.3.3. Mixed-Initiative COA Critics / Advisors (MICCA)

The reasoning component of DEEP was anticipated to reside in three distinct areas: (1) case-based retrieval, and (2) plan adaptation and modification and (3) plan evaluation or assessment. The resources available to the research team precluded serious investigation into these areas. A plan critiquing and advising capability was needed to augment the existing DEEP capability by adding a software component to refine rough, higher-order plans, based on the output of a multiple case-based plan retrieval process, into more detailed, cohesive plans by analyzing aspects such as quality, critical dependencies, possible omissions, undesirable effects, resource constraints, etc. . Coherent plans and context needed to be mapped to a set of critic engines which, in turn, provide their domain expertise back as validation and/or improvements to the plan, making the plan complete and consistent at these lower levels.

Many of the agents in the as-built infrastructure were either nominal or stubbed, with low levels of expertise or ability to handle real-world planning. In addition the goal of a mixed-initiative planning environment, supported by DEEP, had not yet been addressed. To this end, a solicitation was made to do this work under extramural contract and to extend the ARPI CPR schema and blackboard-based architecture to support multi-agent non-deterministic opportunistic reasoning, using case-based reasoning to capture experiences (successes and/or failures) and mixed-initiative critic agents for plan refinement.

This plan critiquing and advising capability component would be used to refine rough, higher-order plans, based on the output of a multiple case-based plan retrieval process, into more detailed, cohesive plans. Review and assessment of specific aspects of the plan at lower levels of detail was desired over a single assessment of the plan as a whole.

A separate research contract, FA98750-10-C-0184, “Mixed-Initiative COA Critic / Advisors”, was initiated. This effort contained three main components: (a) Development of individual critics - addressing scenario interaction, scope of application, initial assessment and categories of expertise; (b) Multi-critic framework - addressing critic interaction and communication, critic control and collaboration, and negotiation and division of labor between human and machine critics; and (c) Enhanced performance - addressing metric identification, explanation and information assurance issues. Execution of the critics was foreseen to be independent of the larger CBR framework, allowing dynamic assembly of a critic community, distributed implementations and sourcing of plans and plan fragments from multiple case-bases.

Issues being addressed under this contractual effort include, but are not limited to: activation of specific critics, de-confliction of critic output, collaboration between critics, plan modification by critics; trust, pedigree and, authority of critics to act; assembly and integration of critics, control and oversight of activated critics, and the necessary run-time support environment, including

interfacing and integration of the critics to both the underlying DEEP architecture and human operators. (Mulvehill, et al., 2011)

Utilizing the aforementioned CPR and blackboard framework, the research team is developing a set of adaptation, critique and assessment plan critics to facilitate the evaluation of plans and partial plans generated by the DEEP-based case-based reasoning framework and adapt them to current conflicts, planning, temporal adaptation and resource scheduling agents to revise candidate plans, and meta-critic agents to select a set of "best" candidate plans for final human review.

Software agents developed under MICCA will fall into the following categories:

- Plan Evaluation and Execution Agents
- Comparison Agents
- Ranking Agents
- Adaptation Agents
- Planning Agents
- Coordination Agents

3.3.3.1. Plan Evaluation Agents and Execution Agents

Evaluation agents are being designed to communicate via the DEEP blackboard and to have their conclusions filtered for execution/selection by both a human and a meta-critic (software agent). The meta-critic agents (comparison and ranking agents) will act as a filter on evaluations and repairs to preclude possible human criticisms. These filtered evaluations and repairs can be presented to the human operator as the best *different options*. Plans will be ranked based on one or more evaluation agent scores along different dimensions using lexicographic preference models (LPMs). An LPM defines an order of importance on the measure variables (criteria) and uses it to make preference decisions. These variables are often domain dependent.

The following plan evaluation agent types have been identified as supporting the evaluation of retrieved/revised plans in any planning domain:

- Executability Evaluation Agent - evaluates the fragility of the plan
- Cost Evaluation Agent - computes the execution cost of the plan
- Risk Evaluation Agent - scores the plan with respect to mission threats
- Policy Evaluation Agent - checks if a plan violates any policies
- Adaptability Evaluation Agent – determines how much adaptation is needed in order to utilize a candidate plan as retrieved from a library of historical plans.

Final implementations of MICCA will contain specialized instances of these evaluation types, e.g., cost evaluations associated with time, fuel, usages of valuable resources.

3.3.3.2. Comparison Agents

The input for this agent is a set of scored plans, although it is not a requirement that every scored plan lead to a pair comparison. A plan is compared to other plans with the same context when all the scores of the plan are available. The comparison agent organizes the scores with respect to the context and the plan. The context of the plan also identifies the lexigraphic preference model (LPM) (Democratic Approximation of Lexicographic Preference Models, 2008) or other model that will be used for comparing plans within the same context. For example, a different LPM might be used for evaluating plans from different domains. Similarly the comparison of revised and retrieved plans might also need different preference models.

The output of this agent is a set of PlanPairComparison objects of the form:

<PlanID1, PlanID2, SourceID, ObjectiveID, StateID, Result, Justification>

Where:

- PlanID1 and PlanID2: Identifiers of the plans that are compared.
- SourceID, ObjectiveID and StateID: come from the Scored Plans that give the scores for these two plans in the same context.
- Result: 0 if the plans are equally good. 1 if the first plan is better, -1 otherwise.
- Justification: The rationale for the result.

3.3.3.3. Ranking Agents

The input for this agent is a set of PlanPairComparison objects. PlanPairComparisons objects induce a total order on the plans that share the same context. The ranking agent then sorts those plans accordingly. The output of the ranking agent is a set of Execution Candidates or Revision Candidates which will be of the form:

<PlanID, SourceID, ObjectiveID, StateID>

Where:

- PlanID: Identifier for the selected plan.
- SourceID, ObjectiveID, StateID: come from the input PlanPairComparison objects.

If the PlanID comes from a retrieved plan (identifiable by the SourceID) then the output will be a RevisionCandidate which will be used by the Adaptation agents. Otherwise the output will be an ExecutionCandidate which will be published to the blackboard and might be stored in the case base and/or passed onto a simulator

3.3.3.4. Adaptation Agents

Several types of adaptation agents might be needed for repairing candidate plans. These will include adaptation agents that are specialized in such areas as: planning, temporal reasoning and resource scheduling, as well as specialized agents that will be designed to handle domain-specific issues, such as route planning, geospatial reasoning and/or motion.

Agent interaction will be managed through a meta-critic or coordination agent. As the adaptation agents take turns to revise the current plan, the coordination agent is responsible for notifying the next agent when it is appropriate to work on the current plan. All adaptation agents will be designed to communicate with the DEEP framework, to pass on information between the agents, and to communicate with the human operator.

There can be several plan adaptation agents ranging from fully domain independent to domain specific. The current design assumes that the plans are hierarchical. As planning agents are envisioned as iterating over every goal task in the objective and relying on subordinate agents or utilities to fix deficient tasks or plans as they are identified.

3.3.3.5. Coordination Agents

Given a candidate plan which might need multiple adaptations, possibly by multiple agents, the challenge becomes one of how the adaptation agents interact with each other to produce coherent plans. The most likely approach for MICCA is to pick one adaptation agent to work on the plan in a given cycle, fix everything it can and then hand it over to another agent. This approach necessitates a meta-critic agent, or coordination agent, that will be responsible for imposing an order on the adaptation agents.

The function of the coordination agent will be to:

- keep track of adaptation agents and their capabilities
- keep track of domains and their needs
- implement a set of agent interaction policies
- implement a set of human interaction policies

In addition to developing the individual critic agents, a framework for supporting the interaction and de-confliction of multiple critics, and a means for exposing the critic activity for human review are also under development. This work is expected to be completed in September 2012.

3.3.4. Case-Based Tactician

This work investigated the use of competitive experience by multiple individuals to develop sensible plans. It utilized an implementation of the Case-based Tactician (CaT) algorithm (Aha, et al., 2006) applied to the StarCraft domain using the DEEP architecture.

This approach is based on a hypothesis, that sensible and relevant solutions, such as a list of suitable candidate actions, can be derived from past planning experience. Genetic Algorithms (GA) were explored to develop an automated capability to learn good plans, and thus be able to select them from a case-base of possible plans. A GA approach is often used when the solution space may be too large or too complex.

There are three major components to the GA: chromosome, fitness function and genetic operations. The chromosome is the generic representation of the solution domain. The fitness function is used to evaluate a goodness of the chromosome. The genetic operations are used to modify and mutate the chromosome. The basic idea is to keep changing and mutating chromosomes and gravitate toward chromosomes that yield better solutions.

The StarCraft domain can be quite complex due to the large number of variables present. The StarCraft solution space was partitioned into a set of states, derived from the build state lattice of the game. The game (or replay of the game) was transcribed as a collection of states making up the chromosomes. The GA fitness function used is described in Figure 15, where the four variables used represent key game parameters easily extracted from game replays.

$$F = S_{pm} + S_{pg} + 2 * S_{pu} + 3 * S_{ku}$$

$$S_{pm} := Mineral\ Score$$

$$S_{pg} := Vaspine\ Gas\ Score$$

$$S_{pu} := Players\ Unit\ Score$$

$$S_{ku} := Enemy\ units\ killed$$

Figure 15: Fitness function for the DEEP-StarCraft GA

Four genes that were used to represent a state, or GA chromosome, are shown in Figure 16: Economy gene, Build gene, Research gene, and Combat gene. The Economy gene specifies when and how many worker drones to build. The Build gene specifies when to build specific types of buildings. The Research gene specifies when to perform research and upgrades on players units. The Combat gene defines the type of units to build, their quantity, and their attack or defend strategy.

The GA operations used were: state crossover 30% of the time, 30% rule replace mutation, 30% biased mutation and 10% randomization (entire chromosome is randomized). The randomization at 10% was too chaotic and was later reduced to 2%.

Resources and priorities prevented a comprehensive development and assessment of this approach. A major impediment to developing an effective GA for case selection was getting an objective measure or a meaningful goodness score. No effective means was available for deriving concise measurements of performance of the DEEP / StarCraft GA approach while the game was running, particularly with the "fog-of-war" option enabled. This was due primarily to the lack of access and availability of objective assessment tools for StarCraft. If future work leads to better means for measuring StarCraft internal performance, this area will be revisited.

Chromosome (representation)

:= State

Economy Gene (number of worker drones)
Build Gene (type of structure to build)
Research Gene (type of research to perform)
Combat Gene (type of combat units to build)

Ex: <<S1, E4, B3, R8, C1, 3, 4>, <S3, ...> ...>

Fitness Function (evaluation metric):

$$F = \begin{cases} \min\left(\frac{GC}{EC} \cdot \frac{M_d}{M_d + M_o} \cdot b\right) & \{d \ \ lost\} \\ \max\left(b \cdot \frac{Md}{M_d + M_o}\right) & \{d \ \ won\} \end{cases}$$

Figure 16: Chromosome representation for the DEEP-StarCraft GA

3.3.5. Simulation of DEEP Generated Plans

AFRL/RIS has had a long-standing relationship with researchers at the Agent Technology Center of the Czech Technical University (CTU) in Prague, Czech Republic. They have been doing research into agent-based computing in distributed adversarial planning. The aim of the project was to investigate problems of adversarial reasoning and planning, and goal-oriented decision making in the presence of other adversarial actors. In contrast to existing approaches, the project aimed to address the problem within the context of complex, asymmetric domains with

properties similar to those found in real world conflict situations (a higher number of actors, asymmetry in an actor's objectives and resources, continuous time and space, etc.)

The pursued approach combined theoretical analysis and practical algorithm development with strong emphasis on empirical evaluation using a multi-agent adversarial behavior simulator test bed. This adversarial planning simulator was integrated as an external component for evaluating candidate operational plans through their simulated execution.

As part of their work, AFRL funded the CTU team to develop tools for automated translation of DEEP plan representation objects into representations executable by their simulator. The simulated execution results were reported back to the user. The results of their work are detailed in the final technical report for FA8655-07-1-3083. (Pechoucek, et al., 2007)

4. Results and Discussion

4.1. Research Platform

A core part of this project was the development of a research platform upon with which the AFRL research team could explore and develop technology to assist commanders and their staff in improving their planning work, drawing on and utilizing past planning output and experience.

As described in this report and in the various references, (Lachevet, et al., 2008), DEEP was successfully implemented as a research platform. It has been utilized for a number of related and spin-off efforts as described in Section 3.3.

The software implementation remains the principle product of the DEEP program. Its usefulness is evidenced by its continued use as an underlying component of ongoing and future research efforts, describe in the following section.

4.2. Future Work

Based on the result of the DEEP project, future research focus has been focused on continuing the Mixed-Initiative COA Critics/ Advisors (MICCA) work as well as the initiation of a new line of inquiry known as Experience Based Adaptation and Replanning (EBAR). The data sources and planning goals for these efforts are drawn from the JAGUAR and StarCraft domains, respectively.

4.2.1. MICCA

Initial work in this area is described in Section 3.3.3. As noted earlier, the technical objective of the MICCA project is to develop software that can be used to aid human operators in adapting and aligning past military plans to meet current objectives and constraints. This work will complement the DEEP technology developed by AFRL. MICCA will develop adaptation, evaluation (for critiquing the plan along one or more dimensions) and assessment routines, in the form of software agents, to assist in transforming *past* plans into viable *current* plans. The long term operational goal of the resulting plan critic advisory system is to provide a commander and his staff a capability to better utilize past experience for planning and generating a new COA in response to a current threat, crisis or adversarial action.

Note that MICCA is avoiding the larger issue of when does plan adaptation devolve into generative planning. The working assumption is that the case bases involved are rich and complete enough to "always" be able to suggest a good first-round planning suggestion. The purpose of this is ensure that the objectives of the effort are attainable and that progress and contribution of the technology is easily measureable against low-complexity baselines, without the effort being drawn into the whole other research area of generative planning.

4.2.2. EBAR

The vision for future Air Force operations is that commanders will be able to continuously monitor and adjust plans during their execution as the situation evolves, and more relevant and timely information becomes available. A more autonomous way of adaptive replanning must be established to provide plausible decisions to the warfighter earlier in the process. Current planning lacks autonomy to provide planning options to operators when plans are deviating from optimal performance as current Air Operations Center processes have too long of a latency window between planning cycles (e.g. many hours). Equally noteworthy is a key finding from *Air Force Technology Horizons* that natural human capacities are becoming increasingly mismatched to the enormous data volumes, processing capabilities, and decision speeds that technologies either offer or demand. (2010)

Planning for military operations is notoriously difficult; initial plans rarely survive first contact with the enemy. The reasons for this are many, including: a significant amount of uncertainty associated with predicting future consequences and events, the complex and real-time nature of these problems forces commanders to make timely decisions without doing a complete search of the domain for planning options, and the adversarial nature of the military domains means that our adversaries will actively perform actions to disrupt our plans.

To achieve these objectives, technology is needed to:

- Devise methods for determining when an executing plan is deviating from its expected performance
- Establish deviation thresholds upon which replanning actions should be undertaken
- Perform rapid, adaptive replanning as necessary to reduce the differential between current state and some desired outcome.

The focus of this work will address the challenge of optimally determining the circumstances under which unforeseen deviations from an executing plan warrant replanning. As military plans are executed monitored plan features will often deviate from what was expected due to incomplete information or actions by an adversary. Once the world state deviates enough the current plan may become non-viable and a new plan will be required. Techniques for deriving robust decision criteria are an open research area to be explored.

The two specific objectives of EBAR are focused on the first two needs listed above:

- Develop a method of measuring the distance between the state variables and between two or more world states. This is critical for determining whether the plan is going as expected, or deviating from expected performance

- Develop a methodology for establishing thresholds for adaptive replanning. Given a method for measuring the deviations in actual plan performance against expected plan performance, the challenge then becomes determining how much of a deviation should be allowed before replanning takes place.

In the real world, intelligence, surveillance and reconnaissance and assessment services exist to provide information from which to build a representation of the current state of the executing plan. In the case of EBAR, this function will be driven by the StarCraft real-time strategy game operating in conjunction with the DEEP blackboard architecture, agents and infrastructure as the default research platform. StarCraft will be used for both generation of cases for historical archiving and retrieval as well as playing forward plans and generating data to be used in assessing the quality of EBAR. Based on the description of the world-state contained in the state variable, the DEEP case-based planning system will retrieve relevant actions to be executed.

Planning under this vision can be viewed as a sequential decision-making process that iteratively modifies states in a state space. Because we are measuring the current world state, this lends itself towards framing the problem as a Markov Decision Process (MDP), represented as a 4-tuple (S, A, T, R), where: S is a set of state variables, A is a set of actions, T is a transition function, and R is the reward function. The state variables comprising S are a set of features representing the world state in which the plan is being executed. (Toward Integrating Feature Selection Algorithms for Classification and Clustering)

The experience base represents individual plans as Markov Chains, containing sequences of states and actions which occurred to reach an objective. Because this is a model-free approach, we do not have complete models of the entire state-space and therefore the probability of an action (a) moving from the current state (s) to the desired state (s') is not 1. This is represented as $\Pr(s,a,s') \neq 1$. Similarly, we are also not undertaking the modeling of the transitional probabilities for transitioning from state to state. Case Based Reasoning allows uncertainty management by drawing upon past experience to choose an action set based on the current state variable. Therefore, when the state variable is sufficiently defined, it can be used to retrieve Markov Chains from the experience base to suggest sequences of states and actions to solve current objectives.

There are two central challenges this effort will address. First, can optimal feature sets for case retrieval and plan monitoring be derived automatically? Features are the sensors and variables that describe the current state of a problem. A concise and complete feature set is critical for managing the size of a search space and ensuring a decision maker, human or synthetic, has enough information to make an appropriate decision. In practice, features are typically manually selected and derived by subject matter experts. Manually selecting features is expensive, error prone, and inflexible. An automated method for feature selection would be much or robust, less costly, and more capable in dynamic domains. Automatic feature selection is an open field of

research where there has been much work in the supervised learning domain (An Introduction to Variable and Feature Selection, 2003) but little in unsupervised learning domains. (Automatic Feature Selection in Neuroevolution, 2005) (Embedded Incremental Feature Selection for Reinforcement Learning, 2011) Advanced in both these areas will be reviewed for application to EBAR.

The second major challenge this work will address is how to automatically determine thresholds for when to replan. As military plans are executed monitored plan features will often deviate from what was expected due to incomplete information or actions by an adversary. Once the world state deviates enough the current plan becomes no longer viable and a new plan is required. Determining what and when is enough is an open research question we want to address. Similar to feature selection, determining thresholds is often a manual process. Past approaches have relied on a commander's intuition to determine thresholds on specified mission parameters. There has been some work in automatic plan monitor thresholding from the robotics community, such as the Rationale Based Monitoring framework. (Rationale-Based Monitoring for Planning in Dynamic Environments, 1998) This approach is promising, but relies on complete models of the domain, which are not readily available, to determine thresholds. Our research will build off of this approach and also investigate using Vector Space Models (Salton, et al., 1975) to determine appropriate distance functions for replanning thresholds.

4.2.3. Robust Coherence

As noted in Section 3.3.1, an important issued identified in the course of the DEEP project was the potential conflict arising from retrieval of multiple past actions, which may not be self-consistent.

Current approaches for synthesizing multiple experiences often involve the use of constraint satisfaction to understand how experiences interact. This sometimes involves representing cases themselves as constraint satisfaction problems (Adaptation Using Constraint Satisfaction Techniques, 1995), or using constraint satisfaction to order actions within a plan. (Merging Strategies for Multiple Case Plan Replay, 1997) Future work in this area will also be using constraint satisfaction techniques, but will be taking a different approach. In our approach, constraints are used to represent the compatibilities and incompatibilities of different perspectives.

Drawing upon ideas from the philosophy of knowledge, our approach to robust coherence can help collaborative partners understand how their epistemic processes differ. Instead of generally disagreeing, different stakeholders can use robust coherence to single out incompatibilities in reasoning and knowledge and narrow the discussion to important issues facing collaborative work. In the future, we will be testing this approach in a variety of knowledge-based domains to

discover how people can better share information collectively in a way that they can agree to and corresponds to the dynamic world.

Our constraint satisfaction algorithm allows us to utilize both soft and hard constraints, the inclusion of which is also a future research area for our coherence approach. The negative constraints produced by the falsification mechanism, for example, could be hard constraints instead of soft. This will enforce correspondence with the world, but may make some of our coherence systems unsolvable. Future efforts will test this hypothesis.

4.2.4. StarCraft

The StarCraft domain will most likely remain the domain of choice for AFRL/RIS internal research programs in case-based reasoning for the near term. As new technologies are identified, investigated and proven on this domain, corresponding extramural efforts will be considered to move out of the laboratory and to apply the results to DoD military domains and problem sets.

The current focus of DEEP on the strategic level of planning is being carried forward into the EBAR research using StarCraft as the domain as that is where experiential reasoning is well suited. There are components of the Starcraft API discussed in Section 3.2.1.2 that allow for the use of "Managers" to handle the tactical level tasks of the game. These managers include:

- Resource Manager –handles the gathering of resources in the game including the production of these resource gatherers.
- Building Manager – alleviates the need of spending time writing algorithms to deal with X & Y placement on the map.
- Attack Manager(s) – applies tactical-level actions and behaviors to specific units.

A high-level view of the DEEP/StarCraft architecture is shown in Figure 17. The central component will continue to be the DEEP Blackboard which easily supports rapid research and testing. The blue objects are the information passed between the StarCraft APIs and DEEP components. The "SCSituation" object is the set of information populated by the StarCraft APIs. This contains information such as map details, unit details, building details, red information, blue locations, etc. The plan feature attribute-value pairs necessary to tracking the world state remain to be defined.

The StarCraft Case-Based Reasoning Agent, SCCBRAgent shown on the left side of Figure 17, represents the case-based reasoning engine currently used in DEEP. This agent uses the current situation to retrieve relevant past experiences (replays) and proposes new plans (actions) based on this information. One of the hard problems remaining to be addressed is when to replan.

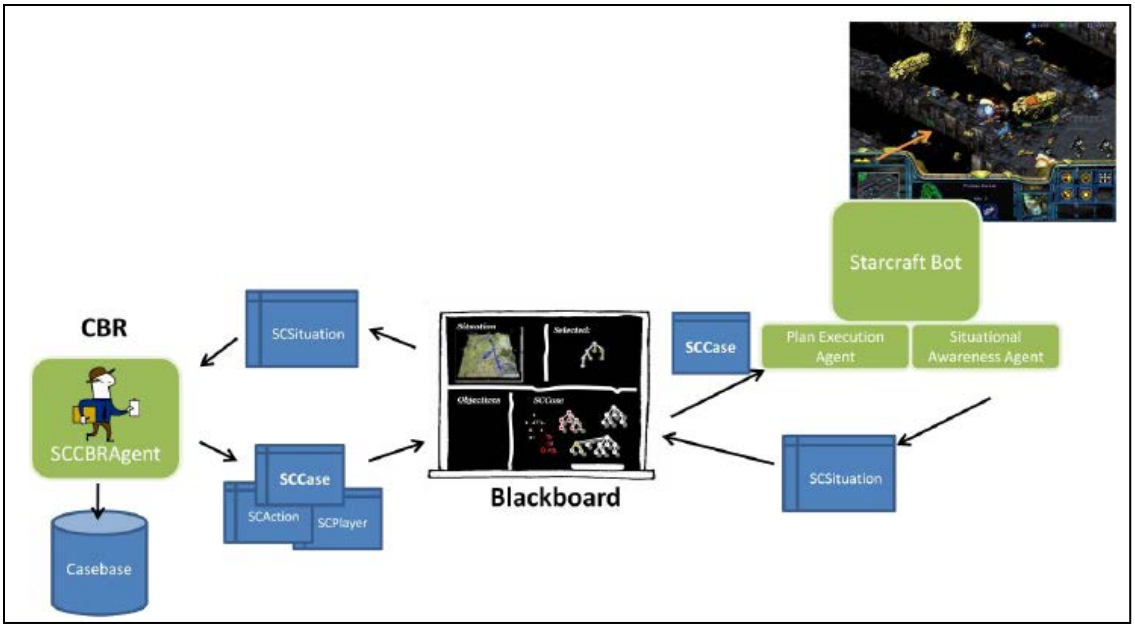

Figure 17: DEEP / StarCraft Architecture

For the EBAR project, the planning system itself is of less interest. The focus of EBAR is to address the challenge of optimally determining the circumstances under which unforeseen deviations from an executing plan warrant replanning. As military plans are executed, monitored plan features will often deviate from what was expected due to incomplete information or actions by an adversary. Once the world state deviates enough the current plan may become non-viable and a new plan is required. The capabilities developed should interact with the planning system on whatever level of complexity the planning system supports. For example, if the planning system supports contingency planning, the developed capability would notify the planning system that some form of contingency planning needs to take place using the latest information. Conversely, if the planning system only supports the creation of a single plan from scratch, the developed capability would notify the system to do a complete re-plan using the latest information of the current situation, or world state.

The duration of the StarCraft game can vary significantly in both time and complexity. Attempting to model the entire (or a previously replayed) game while keeping track of many variables can be overwhelming. Our approach was to utilize Fine State Machines (Howland, 1999) to segment an entire game into a set of smaller manageable units called states. This representation of the game as a state lattice is based on the build tree (or tech tree expansion) of the StarCraft (Broodwars Expansion Pack). Specific buildings enable new features and capacities, e.g., building new types of combat units and buildings, researching new capabilities, etc. The SCReplay (11ht), an open-source java based project, was utilized to read contents of saved StarCraft games. This approach enabled us to effectively reduce the complexity of representing an entire game state through segmentation.

5. Conclusions

DEEP was initiated in 2006 as an internal AFRL Rome Research Site in-house project. The original objective of the DEEP project was to "research, develop, and evaluate technologies for geographically dispersed commanders and senior staff to conduct intuitive, continuous, and distributed planning in the complexity, dynamics, and chaos of forth generation warfare." The proposed approach was to develop a mixed-initiative planning environment to augment the commander's and staff's experience base by drawing on expert knowledge that has been interactively captured and represented in machine understandable form. A spiral development paradigm was envisioned. The first spiral was to leverage readily available components to quickly "breadboard" a basic demonstration of the concept. For the first spiral, the focus was on developing core, individual capabilities rather than integrating the capabilities. The initial goals of the DEEP project were to create a research platform that embodied a complete planning cycle in a distributed architecture, and demonstrate that capability. Latter spirals were expected to enhance the basic functionality and integrate the capabilities.

Focus areas included, but were not limited to: knowledge elicitation and representation for the case-base and episodic memory, analogical reasoning for experience retrieval, principled methods of case adaptation for creating skeleton plans, distributed case base reasoning for plan development and update, and exploratory evaluation of plans by distributed commanders. The products of the research and development effort were identified as demonstration software exemplifying the scientific concepts under study, and papers published in refereed conferences and journals.

The DEEP team successfully met this challenge, providing a solid research platform upon which to build in the following years. As described in this report and in the various references, (Lachevet, et al., 2008), DEEP was successfully implemented as a research platform and has been utilized for a number of related and spin-off efforts.

Although technical advances were made, as the work progressed it became clear that a number of technical issues were beyond the timeframe and resources allocated to the effort. The original, comprehensive DEEP architecture included components for modeling and simulation and for distributed implementations involving networking of multiple DEEP instantiations. Regrettably, work in these areas was not accomplished due to resource limitations and the need to focus on getting the core DEEP functionality coded and running reliably. Subsequently, the research focus was narrowed to concentrate on refining the plan representation scheme, developing appropriate and relevant cases, merging and deconflicting partial plans, and developing protocols and processes for adapting, critiquing and modifying plans.

The software implementation has remained as a demonstration and exploration prototype and has not been hardened or rigorously exercised to a maturity level sufficient to support distribution to

other research communities – it has remained an AFRL/Rome Research Site implementation only. Even within that limited set of applications, DEEP has supported a number of interesting and potential significant research forays, as described in Section 3.3 and will continue to see use in support of the Experience Based Adaptive Replanning and Mixed-Initiative Course-of-Action Critic/Advisors research projects as described in Section 4.2.

6. Bibliography

[Online] UC Berkeley. http://overmind.cs.berkeley.edu/.

[Online] http://www.cs-replay.com.

Adaptation Using Constraint Satisfaction Techniques. **Purvis, Lisa and Pu, Pearl. 1995.** [ed.] M. Veloso and A. Aamody. s.l.: Springer Verlag, 1995. Topics in Case Based Reasoning, Proceedings of the First International Conference on Case Based Reasonin, LNAI Series.

Aha, D, Molineaux, M and Ponsen, M. 2006. Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. *Kunstlichen Intelligenz.* 1 2006, pp. 39-44.

Alberts, D., Hayes, E. 2007. *Planning: Complex Endeavors.* s.l.: Command and Control Research Propgram (CCRP), 2007.

Allen, John, et al. 2004. *Future Command.* s.l. : Defense Advanced Research Projects Agency (DARPA), 2004. DARPA ISAT Study.

An Introduction to Variable and Feature Selection. **Guyon, I and Elisseeff, A. 2003.** 2003, Journal of Machine Learning Research, Vol. 3, pp. 1157-1182.

Argelich, J, et al. Fourth Max-SAT Evaluation. *Max-SAT 2009.* [Online] [Cited: October 20, 2011.] http://www.maxsat.udl.cat/09/index.php?disp=organizers.

Automatic Feature Selection in Neuroevolution. **Whiteson, S, Stone, P and Stanley, K. 2005.** 2005. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 05). pp. 1225-1232.

Bichindaritz, I. 2006. Memory Organization as the Missing Link Between Case Based Reasoning and Information Retrieval in Biomedicine. *Computational Intelligence.* 2006, Vol. 22, pp. 148-160.

Braun, G. 2006. *AFFOR Command and Control Enabling Concept - Concept 2.* s.l.: USAF/A5XS, 2006. Internal.

Carozzoni, Joseph, et al. 2008. *Distributed Episodic Exploratory Planning.* s.l.: Air Force Research Laboratory, 2008. Interim Technical Report. AFRL-RI-RS-TR-2008-279.

Coherence as Constraint Satisfaction. **Thagard, P., Verbeurgt, K. 1998.** 1, 1998, Cognitive Science, Vol. 22, pp. 1-24.

Collaborating with Multiple Distributed Perspectives and Memories. **Ford, Anthony and Mulvehill, Alice. 2009.** 2009. Proceedings of the International Workshop on Social Computeing, Behavioral Modeling and Prediction.

Corkill, D. 1991. Blackboard Systems. *AI Expert.* 1991, Vol. 6, 9, pp. 36-38.

Creating and Capturing Expertise in Mixed-Initiative Planning. **Ford, A., Carozzoni, J. 2007.** 2007. Proceedings of the 12th International Command and Control Research and Technology Symposium.

Crumb, F. 2003. AFRL-Rome Awards contracts for JAGUAR Program. *Defense AT&L.* October 24, 2003.

Defeating Novel Opponents in a Real-Time Strategy Game. **Molineaux, M, Aha, D and Ponsen, M. 2005.** 2005. Proceedings of the IJCAI 2005 Workshop on Reasoning, Representation, and Learning in Computer Games.

Democratic Approximation of Lexicographic Preference Models. **Yaman, F, et al. 2008.** 2008. Proceedings of the ICML 2008.

Distributed Planning in a Mixed-Initiative Environment Collaborative Technologies for Network centric Operations. **Destafano, C., Lachevet, K. and Carozzoni, J. 2008.** 2008. Proceedings of the 13th International Command and Control Research and Technology Symposium.

ELOP-DEEP. **Staskievich, G. 2011.** s.l. : (Unpublished), 2011.

Embedded Incremental Feature Selection for Reinforcement Learning. **Wright, R, Yu, L and Loscalzo, S. 2011.** 2011. Proceeding of the 2011 ICAART.

Engelmore, Robert and Morgan, Tony. 1988. *Blackboard Systems.* 1988.

Expectation Failure as a Basis for Agent-Based Model Diagnosis and Mixed-Initiative Model Adaptation During Anomalous Plan Execution. **Mulvehill, A, Benyo, B and Cox, M, Bostwick, R. 2007.** Hyderabad, India : AAAI Press, 2007. Proceedings of the 20th International Joint Conference on Artificial Intellligence. pp. 489-494.

Eye-Tracking Analysis of User Behavior in WWW Search. **Granka, L, Joachims, T and Gay, G. 2004.** Sheffield, South Yorkshire, UK : ACM, 2004. SIGIR 2004.

Hasseler, G. 2010. *Adapting Stochastic Search for Real-Time Dynamic Weighted Constraint Satisfaction.* s.l.: State University of New York, 2010. Master's Thesis.

Howland, G. 1999. *A Practical Guide to Building a Complete Game AI: Vol I.* s.l. : GameDev.net, 1999.

Hughes, C. and Hughes, T. 2003. *Parallel and Distributed Programming Using C++.* s.l. : Pearson Education, 2003.

Improving Plan Adaptation Process Through Semantic Technologies. **Staskevich, Gennady and Carozzoni, Joseph. 2009.** 2009. Proceedings of the 14th International Command and Control Research and Technology Symposium.

Kolodner, Janet. 1993. Case-Based Reasoning. *Morgan Kaufmann Series in Represenation and Reasoning.* s.l.: Morgan Kaufmann, 1993.

Lachevet, Kurt, Kaczynski, D. and Rogers, G. 2008. *Distributed Episodic Exploratory Planning.* s.l.: Air Force Research Laboratory, 2008. Contract FA88750-07-C-0176, AFRL-RI-RS-TR-2008-322.

Merging Strategies for Multiple Case Plan Replay. **Veloso, Manuela M. 1997.** [ed.] D. B., Plaza, E. Leake. 1997. Proceedings of the 2nd International Conference on Case-Based Reasoning.

MiniMaxSat: A New Weighted Max-SAT Solver. **Heras, F., Larrosa, J., Oliveras, A. 2008.** Guangzhou, China: s.n., 2008. Theory and Applications of Satisfiability Testing SAT 2008, 11th international Conference.

Mission Assurance in a Distributed Environment. **Destefano, Chad and Clark, Thomas. 2009.** 2009. Proceedings of the 14th International Command and Control Research and Technology Symposium.

Mixed-Initiative Planning in a Distribued Case-Based Reasoning System. **Lachevet, Kurt. 2009.** 2009. Proceedings of the 14th International Command and Control Research and Technology Symposium.

Mulvehill, A, Benyo, B and Yaman, F. 2011. *MICCA Consolidated Design Document.* s.l.: (unpublished), 2011. Contract FA8750-10-C-0184.

Mulvehill, A., Benyo, B. 2011. *Agent-Based Model Diagnosis and Repair.* s.l.: Air Force Research Laboratory, 2011. Contract FA8750-10-C-0149, AFRL-RI-RS-TR-2011-139.

Mulvehill, A., Deutsch, S. and Rager, D. 2007. *Case-Based Reasoning for DEEP: Observations and Recommendations.* s.l.: BBN Technologies, 2007. BBN-12-071.

Pease, A. 1998. *Core Plan Representation.* 1998.

Pechoucek, M, et al. 2007. *Agent-Based Computing in Distributed Adversarial Planning.* Gerstner Laboratory, Agent Technology Center, Czech Technical University. s.l. : U.S. Air Force, 2007. Final. Contract FA8655-07-1-3083.

Popper, Karl. 1963. *Conjectures and Refutations: The Growth of Scientific Knowledge.* s.l.: Rutledge, 1963.

Rationale-Based Monitoring for Planning in Dynamic Environments. **Veloso, M, Pollack, M and Cox, M. 1998.** Pittsburgh, PA: s.n., 1998. Proceedings for the 4th International Conference on AI Planning Systems. pp. 171-179.

2010. *Report on Technology Horizons: A Vision for Air Force Science and Technology During 2010-2030.* AF/ST. s.l.: U.S. Air Force, 2010. AF/ST-TR-10-01.

Robust Coherence: An Approach to Multi-Agent Experience-Based Planning. **Ford, Anthony. 2009.** 2009. 2009 AAAI Spring Symposium on Technosocial Predictive Analytics.

Salton, G, Wong, A and Yang, C. S. 1975. A Vector Space Model for Automatic Indexing. *Information Retrieval and Language Processing.* 1975, pp. 613-620.

Semantic Adaptation AGEnts (SAAGE). **Staskevich, Gennady and Carozzoni, Joseph. 2009.** 2009. Proceedings of the Knowledge Systems for Coalition Operations Conference.

Semantic Interoperability in Distributed Planning. **Staskevich, Gennady, et al. 2008.** 2008. Proceedings of the 13th International Command and Control Research and Technology Symposium. Post-Printed as AFRL-RI-RS-TP-2008-6.

Synthesizing Disparate Experiences in Episodic Planning. **Ford, Anthony and Lawton, James. 2008.** 2008. Proceedings of the 13th International Command and Control Research and Technology Symposium.

Thagard, P., Milligram, E. 1995. Inference to the Best Plan: A Coherence Theory of Decision. [ed.] A., Leake, D. Ram. *Goal-Driven Learning.* 1995, pp. 439-454.

The Coherence Theory of Empirical Knowledge. **Bonjour, Lawrence. 1976.** 5, 1976, Philosophical Studies, Vol. 30, pp. 281-312.

Toward Integrating Feature Selection Algorithms for Classification and Clustering. **Liu, H and Yu, L.** 4, IEEE Transactions on Knowledge and Data Engineering (TKDE), Vol. 17, pp. 491-502.

Using Latent Semantic Analysis to Improve Information Retrieval. **Dumais, S, et al. 1988.** New York, NY : ACM, 1988. Proceedings of CHI'88: Conference on Human Factors in Computing. pp. 281-285.

Symbols, Abbreviations and Acronyms

AF/A5 – Air Force Plans Office
AFFOR – Air Force Forces
AFRL – Air Force Research Laboratory
AFRL/RI – Air Force Research Laboratory Information Directorate
AFRL/RIS – Air Force Research Laboratory Information Systems Division
AI – Artificial Intelligence
AOC – Air Operations Center
API – Application Programming Interface
ARPI – Advanced Research Projects Agency (ARPA) Rome Planning Initiative
API – Application Programming Interface
ATO – Air Tasking Order
BBUID – Blackboard Unique Identifier
BTHAI – A StarCraft Broodwar "bot"
BWAPI – Brood War API
C2 – Command and Control
CaT – Case-based Tactician
CBR – Case Based Reasoning
COA – Course of Action
CPR – Core Plan Representation
CSP – Constraint Satisfaction Problem
CTU – Czech Technical University
DARPA – Defense Advanced Research Projects Agency
DEAR – Distributed Episodic Analogical Reasoning
DEEP – Distributed Episodic Exploratory Planning
DTIC – Defense Technical Information Center
DWCSP – Dynamic Weighted Constraint Satisfaction Problem
EBAR – Experience Based Adaptation and Replanning
ELOP – ELementary OPtimization
GA – Genetic Algorithm
JAGUAR -- Joint Air Ground Unified Adaptive Replanning system
LPM – Lexicographic Preference Model
LSA – Latent Semantic Analysis
MDP – Markov Decision Process
MICCA – Mixed-Initiative COA Critics / Advisors
NCO – Network Centric Operations
SAAGE – Semantic Adaptation AGEnts
SAT – SATisfiability problem
SVD – Singular Value Decomposition

UID – Unique Identifier
USAF – United States Air Force
WCSP – Weighted Constraint Satisfaction Problem
WWW – World Wide Web
XML – eXtensible Markup Language